Techniques for Content-based Image Characterization in Wavelets Domain

Georgios Voulgaris

University of Glamorgan
Prifysgol Morgannwg

FACULTY OF ADVANCED TECHNOLOGY
DEPARTMENT OF COMPUTING
AND MATHEMATICAL SCIENCES

A thesis submitted in partial fulfilment of the requirements of the University of Glamorgan/Prifysgol Morgannwg for the degree of Doctor of Philosophy

June 2008

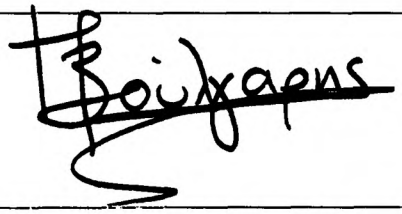
Table of Contents

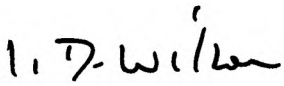
Table of Figures

Table of Algorithms

Certificate of Research

This is to certify that, except where explicit reference is made, the work presented in this thesis is the result of the investigation undertaken by the candidate.

Candidate: 

Director of Studies: 

Acknowledgements

I would like to express sincere thanks to my supervisors at the University of Glamorgan, Dr. I.D. Wilson, Prof. J.A. Ware, Dr. P. Jarvis for the support, guidance and constructive criticism they have provided in the course of this research project. Special thanks to Prof. Jianmin Jiang of University of Bradford and Dr. Mike Reddy of University of Newport, both formerly of University of Glamorgan, for their input and guidance.

Special thanks also to my current employer, VisionMobile Ltd, and in particular Dr. Andreas Constantinou, director of the company and personal friend, for his invaluable support and patience during the completion of this journey.

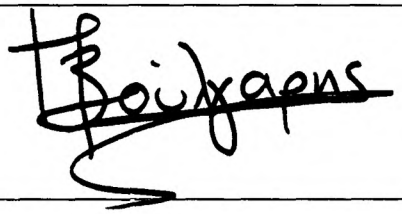
Thanks to former fellow research student Dr. Andrew Armstrong for his camaderie and support.

Finally, thanks to my family, my mother Grammatiki and brother Antonis, for their love and support during this effort.

Declarations

This is to certify that neither this thesis nor any part of it has been presented or is being currently submitted in candidature for any degree other than the degree of Doctor of Philosophy of the University of Glamorgan.

Candidate:



Abstract

Techniques for Content-based Image Characterization in Wavelets Domain

Georgios Voulgaris

Doctor of Philosophy

This thesis documents the research which has led to the design of a number of techniques aiming to improve the performance of content-based image retrieval (CBIR) systems in wavelets domain using texture analysis. Attention was drawn on CBIR in transform domain and in particular wavelets because of the excellent characteristics for compression and texture extraction applications and the wide adoption in both the research community and the industry. The issue of performance is addressed in terms of accuracy and speed.

The rationale for this research builds upon the conclusion that CBIR has not yet reached a good performance balance of accuracy, efficiency and speed for wide adoption in practical applications. The issue of bridging the sensory gap, which is defined as *"[the difference] between the object in the real world and the information in a (computational) description derived from a recording of that scene."* has yet to be resolved. Furthermore, speed improvement remains an uncharted territory as is feature extraction directly from the bit-stream of compressed images.

To address the above requirements the first part of this work introduces three techniques designed to jointly address the issue of accuracy and processing cost of texture characterization in wavelets domain. The second part introduces a new model for mapping the wavelet coefficients of an orthogonal wavelet transformation to a circular locus. The model is applied in order to design a novel rotation-invariant texture descriptor. All of the aforementioned techniques are also designed to bridge the gap between texture-based image retrieval and image compression by using appropriate compatible design parameters. The final part introduces three techniques for improving the speed of a CBIR query through more efficient calculation of the L_1-distance, when it is used as an image similarity metric. The contributions conclude with a novel technique which, in conjunction with a widely adopted wavelet-based compression algorithm, extracts texture information directly from the compressed bit-stream for speed and storage requirements savings. The experimental findings indicate that the proposed techniques form a solid groundwork which can be extended to practical applications.

Thesis Outline

Thesis structure

Chapter 1: Provides the background and rationale for the research.

Chapter 2: Is a critical review of the bibliography in the area of low-level feature extraction research focusing on texture in wavelets domain. Contributions on similarity calculation are also discussed.

Chapter 3: Introduces three new wavelet-based feature extraction methods based on quad-trees and the autoregressive occurrence model.

Chapter 4: Presents a novel approach for rotation invariance in wavelets-based texture analysis.

Chapter 5: Describes two techniques for feature signature pre-filtering in order to improve the speed of signature matching process. Another technique is presented which allows the extraction of feature information directly from the compressed bit stream.

Chapter 6: Concludes with a critical discussion of the work described in this thesis. Furthermore, a number of suggestions are made for future extension of this work and new ideas inspired by the work presented in this thesis are presented.

Aims of research

The general aim of this research is to provide the means for improving accuracy and speed of content-based image indexing, retrieval and classification in wavelets domain. In particular, this research sought to:

- Suggest new and build upon existing techniques for improving the accuracy of content-based image retrieval and classification; ideally these should be compatible and capable of being extended to compressed domain.
- Suggest methods for improving the speed of CBIR/CBIC systems;
- Suggest methods for feature extraction directly from the compressed-bit stream without decompression while ensuring that accuracy and speed remain unaffected.

Chapter 1: Introduction

1.1 Prologue

Information Technology experts often refer to Moore's Law, quoting the prediction by Gordon Moore (Moore 1965), co-founder of chip manufacturer Intel, that the density of transistors per integrated circuit would grow exponentially over time (double approximately every two years). Scientists have expressed scepticism on how long Moore's Law will hold true due to the physical limits imposed by component miniaturization (Stokes 2008). Nowadays, manufacturers still succeed in proving those claims wrong by introducing new manufacturing methods, materials and chip architectures which extend Moore's law for at least another decade (Greene 2007; Markoff 2007). Practically, this growth is one of the major factors for the increased processing power, increased system memory capacity and the miniaturization of electronic devices. Digital imaging is one of the fields that has benefited the most from this wealth of resources.

Nowadays digital cameras with the capability to capture photographs and video of multi-megapixel sizes are commonplace. More importantly, general purpose home computers have enough storage capacity and processing power in order to easily archive and manipulate multimedia content such as digital images, music and video.

Digital image databases have many applications beyond personal photograph collections, such as: preservation of art collections, press archives, security system cameras archives and medical imaging. Naturally, one of the biggest databases is the World Wide Web (WWW or Web), containing images in the order of magnitude of several millions. In 1997 the number of images on the Web was estimated to be between 10 and 30 millions (Sclaroff, Taycher et al. 1997). Search engine Google now counts over 1 billion images! Having the ability to store such a large volume of non-textual data yields the question: "how can one index such data for efficient and accurate retrieval?" A consequence of the openness of the Web is that the increasing quantity of information on the Web also increased the variation in terms of relevance and quality of the content. Therefore, the retrieval problem has changed over the years from "Is there any relevant information on the web?" to "how can I locate the

most relevant information within an acceptable time frame?" hence bringing the issues of accuracy and efficiency at the forefront (Kherfi, Ziou *et al.* 2004).

The first part of this chapter (Sections 1.2-1.5) provides a brief outline of Content-Based Image Retrieval (CBIR), thus setting the scene for the second part (Sections 1.6-1.8), which outlines some of the yet unresolved issues in the CBIR research practice. The second part concludes with the formulation of the basic research aims inspired by a subset of the unresolved issues.

1.2 Describing images

1.2.1 Using text

In database applications textual fields are commonly employed for the description of characteristic properties of the contents (Eakins and Graham 1999; Bashir, Khanvilkar *et al.* 2003). Bringing this to the context of an image database, one could identify a number of characteristics which can uniquely and intuitively describe the content of an image. These characteristics can then be encoded into text fields. For example, in order to describe a collection of images showing human faces those fields may include: hair colour, eye colour, facial shape, eye shape *etc.* Such data can be embedded into the image file as meta-data or stored in a separate database with a reference pointer to the original image file.

The main drawbacks of textual fields in the context of image characterization are:

- Different people might have different perceptions of a specific characteristic (*e.g.* not all people perceive colours in the exact same way), (Seloff 1990; van der Starre 1995)
- Different people might use different phrasing and words to describe the same characteristic, (Keister 1994)
- The use of textual keys makes the initial set-up as well as the maintenance of the database difficult, error prone and time consuming since it has to be performed manually, (Ogle and Stonebraker 1995)
- In order to minimize the effect of a) and b), the meta-data would occupy a lot of storage space in an attempt to accommodate many alternatives.

To this end, content-based image characterization offers an alternative to textual descriptors. In content-based image characterization all the necessary content information is extracted by appropriate mathematical manipulation of the image raw data.

1.2.2 Using Low-level Features

Starting from the pixel representation of a digital image, the basic set of image features that can be easily derived mathematically are luminance (greyscale) and chrominance levels. A pixel of a digital image is numerically described by the magnitudes of three primary colours: Red, Green and Blue. This description is also known as the RGB colour space. Luminance and chrominance levels can be obtained by mapping of the RGB colour space to the YUV colour space using an appropriate transformation matrix. In the YUV colour space the letter Y stands for luminance and the letters U, V stand for chrominance levels. Excellent definitions of the different colour spaces can be found in (Jain 1989). Luminance and chrominance yield information about edge (or, from a different perspective, smoothness) and colour image content.

From this basic content information a low-level feature set can be derived which includes colour, texture and shape. Early research for the design of low level features date back in the 1970s (Julesz 1975; Weszka and Deya 1976; Haralick 1979). Because of the straightforward methodology to translate such methods into algorithms, image content characterization based on low-level features is very popular in image processing research. Not all systems use an exhaustive set of low-level features. The selection is either dictated by the application, as it has been proven that feature performance is application-dependent (Deselaers, Keysers *et al.* 2008), or simply by user specification.

1.2.2.1 Colour

Colour is an important low-level feature because it is also a major discriminative factor for the human visual system (Wu, Kankanhalli *et al.* 2002). A histogram is a common approach for the quantification of the colour content of an image. The histogram is a discrete function of the number of pixels that belong to a colour (based on pre-defined colour quantization) over the total number of pixels

comprising the image. This represents, in terms of statistics, the probabilities of occurrence of individual colours.

1.2.2.2 Texture

Texture refers to the repetition of basic patterns that form homogeneous areas in an image. The basic texture patterns are called texels (Jain 1989). A texel can be periodic, quasi-periodic or random in natural images (Mandal, Idris et al. 1999). Properties such as edge information and patterns are extracted by the luminance or grey-scale channel of an image. As part of a CBIR/CBIC system, texture analysis may involve one or more discrete stages, depending on the application, as follows (Materka and Strzelecki 1998):

- *Feature extraction* is the computation of a characteristic of a digital image which quantitatively describes its texture properties.
- *Texture classification* is to establish to which of a predefined set of physically distinct classes (*e.g.* wood and sand) a uniform texture region belongs.
- *Texture discrimination* or *segmentation* is the division of an image into regions of perceptually uniform textures.
- *Shape from texture* is to recreate three-dimensional surface geometry from texture information.

Texture analysis and modelling can be categorized into three different types (Tuceryan and Jain 1993): structural, statistical and spectral. A detailed account of these three categories is given in Chapter 2.

1.2.2.3 Shape

Shape can be categorized into global and local. Global properties are extracted from the entire shape, for example, roundness, circularity, central moments and eccentricity (Tegolo 1994). Local features are derived by partial analysis of a narrower section of a shape, including dimensions and directionality of adjacent boundary sections (Eakins and Graham 1999), points of curvature, corners and turning angle (Tegolo 1994).

1.2.2.4 Applications

Applications of image retrieval and classification based primarily or exclusively on low-level features include, but are not limited, to:

- Geographical Information Systems need to process very large images and segment them according to landscape features for mapping purposes (Zuyuan and Boesch 2007);
- Computer Aided Diagnosis systems processes such as Electrocardiograms (ECG), Magnetic resonance imaging (MRI), Computerized Thermal Imaging (CTI), Ultrasound *etc.* data, in order to automatically detect anomalies and defects (Muller 2004; Oliveira, Cirne *et al.* 2007);
- Manufacturing automated quality control systems use computer vision algorithms in order to check that the condition of the end product meets the requirements (Qiao, Murtagh *et al.* 2004);
- Biometrics, *i.e.* technologies that measure and analyze human body characteristics, such as fingerprints, eye retinas and irises, voice patterns, facial patterns and hand measurements, for authentication purposes. (Nabti, Ghouti *et al.* 2006);
- Censored/explicit content filtering for internet browsers (Shih, Lee *et al.* 2007).

1.3 Querying an image database

Eakins (Eakins and Graham 1999) has identified and documented the different types of CBIR queries from a user's perspective. The model has also been revisited and adopted recently by (Liu, Zhang *et al.* 2007), as follows:

a. Low-level feature queries: Images that contain one or more dominant colours, or a particular shape. Images that look like other images.
b. Combinatory feature queries: Images that contain specific object types (houses, tree *etc.*). Images of specific objects or people (*e.g.* Big Ben or Brad Mehldau),
c. Abstract attribute queries: Images of named events (*e.g.* carnival) or images of emotional significance (*e.g.* containing happy people.),
d. Image Acquisition data: For example, date/time that the image was taken, digital image format type, image size.

Low-level feature queries can be easily formulated in algorithmic form, hence they are more appropriate for automated computation. The metrics of this type of queries are objective since they can be derived without any external information. The

problem with low-level feature queries is that they are generally restricted to cases where this type of criteria is sufficient *i.e.* specialized applications where the degree of similarity between two images can be accurately represented by the colour, texture or shape features. As is discussed in Section 1.6.3, recent studies attempt to overcome this issue by introducing semantic annotation and analysis techniques. Low-level feature queries are often applied in systems where the context of the image is known, for example biometrics, face recognition, trademark registration.

Combinatory queries fuse information gathered from a set of low-level features in order to extract information regarding, for example, object type, or, the identity of someone or something. However, this is quite difficult to achieve for practical applications as it usually requires extensive system knowledge for identification of the image context and individual objects. The same restrictions apply to abstract attribute queries. For this reason, combinatory and abstract attribute queries are said to represent *semantic queries.* Semantic queries require additional knowledge regarding the context of an image to that which is available from low-level feature analysis.

Image acquisition data are very important and commonly used for image retrieval. It relies exclusively on meta-data that are usually embedded into the image file. Acquisition data can often be the only known attribute of a query, *i.e.* "when was the image shot or downloaded?"

The ability to extract visual content information automatically via mathematical manipulation of digital images and the requirement for low-level feature queries gave birth to the research areas of Content-based Image Indexing, Retrieval (CBIR) and Classification (CBIC).

1.4 CBIR/C architecture

Despite the different outcomes of the two basic applications of image content analysis, retrieval and classification, their internal mechanics are largely similar. In this section the typical stages of each process are outlined with references to some design issues. These issues are elaborated in Section 1.6.

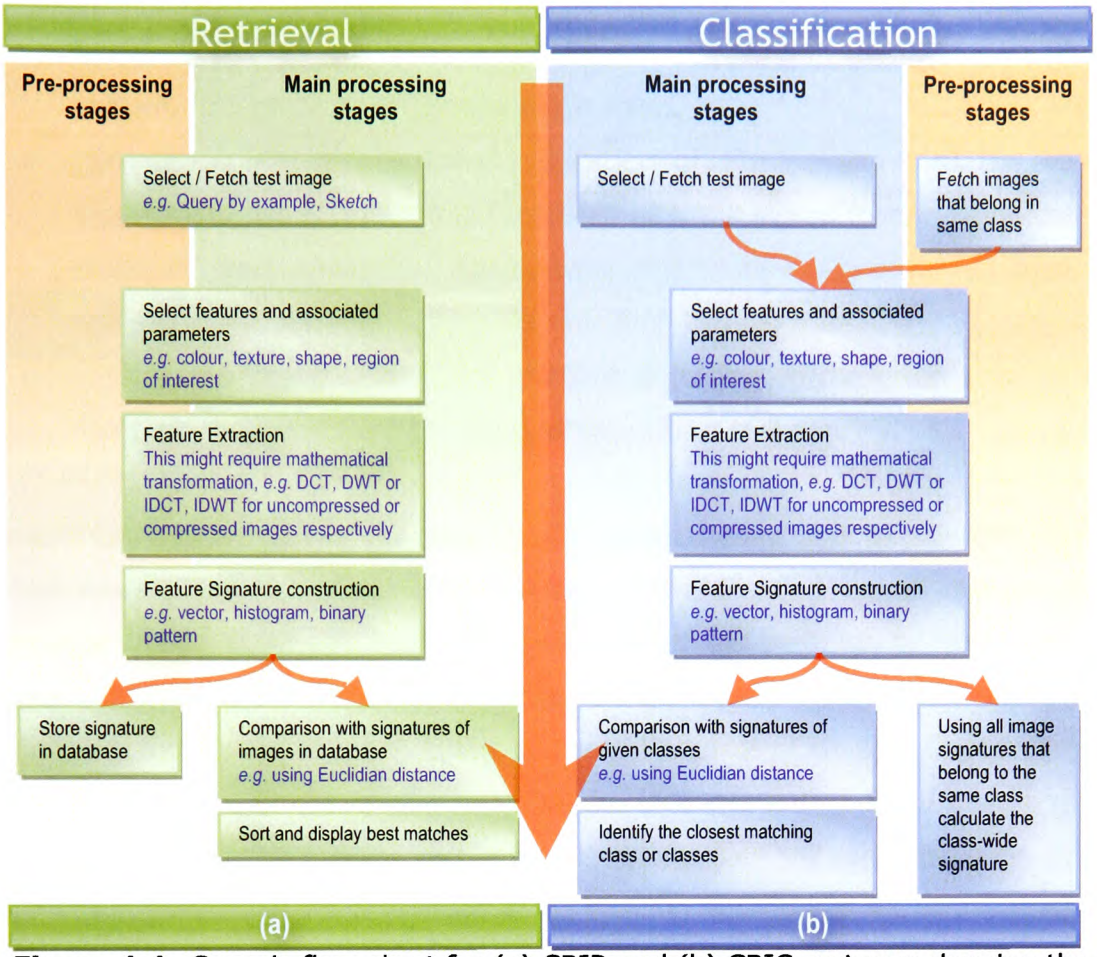


Figure 1.1: Generic flow chart for (a) CBIR and (b) CBIC systems; drawing the threads of Sections 1.4.1 and 1.4.2 together.

1.4.1 Content-based Image Retrieval (CBIR) Systems

The architecture of a CBIR system typically consists of (Figure 1.1a):

1. Source selection: Query image which is either fed to the system (*i.e.* scanned), selected from a given set, or roughly sketched using a digitizer.
2. Selection of features to be taken into consideration: For example, colour or both colour and texture and the extent to which each feature contributes to the signature.
3. Extraction of feature information: Low-level features are extracted and represented in quantitative form through mathematical manipulation of the image data.
4. Feature signature creation: the collected data that represent the selected feature or features are aggregated and transformed into a format that

facilitates storage and comparison. Typical formats are: matrix/vector, binary pattern, histogram or single numerical value.

5. Comparison with images stored in database or with learning rules: the feature signature of the test image is compared with the stored feature signatures to establish level similarity. Alternatively the feature signature is used for training of the retrieval system via specified learning rules.
6. Sorting and listing of the best matches according to the given criteria: The more similar images are retrieved as the output of the query.

Generic CBIR systems are plagued by the inconsistency and peculiarities of the human visual perception (Neumann and Gegenfurtner 2006). If two people are given the same image they will both describe it using different words and they will probably identify different dominant features. Moreover, consider an experiment where both testers are asked to select and rank, in terms of similarity to the original specimen, ten images out of the same bunch. It is more than likely that they will select different sets of images and almost certainly they will not rank them in the same order. In Section 1.6 a number of the existing approaches for overcoming this issue are outlined.

The human visual perception issue is not encountered in application-specific systems where the semantic space of the retrieval is confined. To illustrate the concept, consider the case of a fingerprint recognition system. In fingerprint recognition, specific edge-derived features are used in order to identify similar fingerprints. The reason why fingerprint recognition can achieve such high levels of accuracy is that several parameters of the specimen image conform to very strict specifications. For instance, colour information can be discarded since only edge information is of interest; maximum contrast can be automatically adjusted since no background exists; scale and orientation can be easily normalized. In this way, the data set is homogeneous and the feature extraction algorithm can be formulated with very specific rules. In many cases, the nature of application-specific systems also makes evaluation more straightforward. This is true when the ground-truth (*e.g.* only one fingerprint is a correct match) can be established in a much more deterministic way than generic retrieval systems.

Similarly to the distinction between Generic and Application-specific CBIR, Smeulders (Smeulders, Worring *et al.* 2000) categorized the two respective image domains as *narrow* and *broad* giving the following definitions:

> *"A narrow* [image] *domain has a limited and predictable variability in all relevant aspects of its appearance. [...] In a narrow domain, one finds a limited variability of the content of the images*".

> *"A broad* [image] *domain has an unlimited and unpredictable variability in its appearance even for the same semantic meaning. [...] In broad domains, images are polysemic and their semantics are described only partially."*

1.4.2 Content-based Image Classification (CBIC) Systems

CBIC systems are divided into two major categories, namely: *supervised* and *unsupervised*. *Supervised* systems require a-priori training in order to establish the different feature classes. *Unsupervised* systems dynamically create the different image classes.

In Pattern Recognition terms, supervised learning techniques require that the class labels are known beforehand, whereas unsupervised learning techniques require that the labels must be inferred from the extracted feature information. An example of the latter is image segmentation where pixels need to be grouped into homogeneous clusters in order to split the image to regions. Image segmentation is commonly used in GIS systems for map regions delineation (Zuyuan and Boesch 2007).

As it can be seen in Figure 1.1b, the architecture of a CBIC system resembles CBIR to a large extent. The main difference is that the output of a CBIC system is either the class or the classes to which the specimen image belongs whereas the output of a CBIR system is typically a list of similar images from the database.

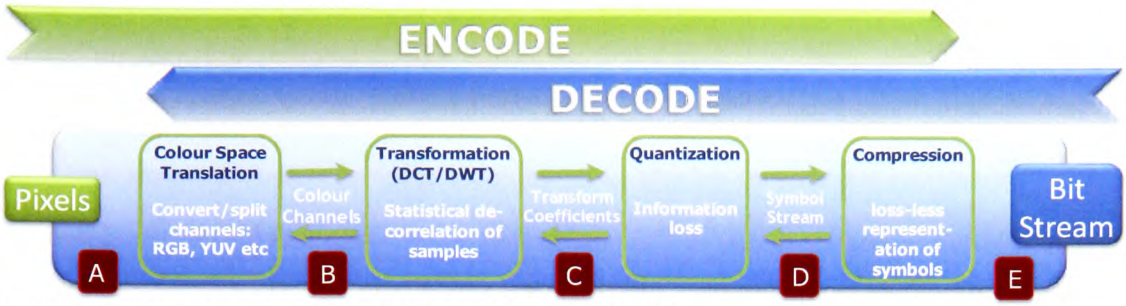


Figure 1.2: A generic model of a transform codec

1.5 Low-level features extraction in compressed domain

Compression is central to image processing for efficient storage and network bandwidth utilization. Figure 1.2 depicts the processing stages of a typical image compression codec. Most compression codecs include one stage prior to encoding, which transforms raw pixel data into a more compact form. This typically involves the use of a mathematical transformation, such as the Discrete Cosine Transform (DCT) and the Discrete Wavelet Transform (DWT). The transformation stage statistically de-correlates the data, making it easier to decide what can be discarded without or with little loss in quality.

Using pixel-based techniques to perform feature extraction means that the image has to be processed either before compression or after full decompression. It is therefore desirable, with respect to Figure 1.2, to "shift" feature extraction processing as close to point E, which represents the compressed bit stream. Ideally, an algorithm would extract all necessary features from the compressed data stream without requiring any of the decompression stages. However, in existing research, feature extraction in compressed domain usually refers to processing transform coefficients (see point C in fig. 1.2).

Working on transformed coefficients instead of raw pixel data has another major advantage. The aforementioned mathematical transformations produce frequency response or spatial-frequency response maps. In the context of imaging, high frequency components represent sudden changes of intensity (in other words, sharp edges or rough surfaces), whereas, low frequency components represent smooth areas. This is particularly useful in the formulation of low-level features such as

shape and texture, which rely solely on detecting surface (or intensity in two-dimensional terms) anomalies.

From the above it becomes clear that there is strong motivation to perform feature extraction in transform domain. The two most widely deployed mathematical transformations in image processing are the Discrete Cosine Transform (DCT) and the Discrete Wavelet Transform (DWT), which are outlined in the following sections.

1.5.1 Discrete Cosine Transform

Based on the Discrete Fourier transform, Discrete Cosine Transform employs real sinusoidal basis functions (Jain 1989). For most natural images compaction efficiency of DCT is very high (reportedly near the optimum Karhunen-Loeve transform). Therefore, DCT is employed by many international image and video standards, namely the Joint Photographic Experts Group (JPEG), Moving Picture Experts Group (MPEG) 1 and 2 and H.261/H.263. The widespread use of JPEG for image compression on the internet has brought DCT to the focal point of image indexing and retrieval research community for years.

1.5.2 Discrete Wavelet Transform

Since the early 1990s, the Discrete Wavelet Transform (DWT) and in particular Subband Coding have become very popular in image coding and indexing (Mandal, Idris et al. 1999; Smeulders, Worring et al. 2000; Datta, Joshi et al. 2008).

The Wavelet Transform is a mechanism which transforms an image from the pixel-domain to the frequency-spatial domain. The information that is conveyed in a pixel-domain representation is location (relative to some reference point) and intensity, whereas in frequency-spatial domain the information is location, intensity and frequency. The following example gives an indication of how to interpret different frequencies in the context of DWT. If the pixel-domain intensity values are considered as a discrete 2-D sample then sudden changes of intensity (sharp edges) correspond to high frequency components in DWT domain whereas areas of uniform intensity (smooth areas) corresponds to low frequency components in DWT domain.

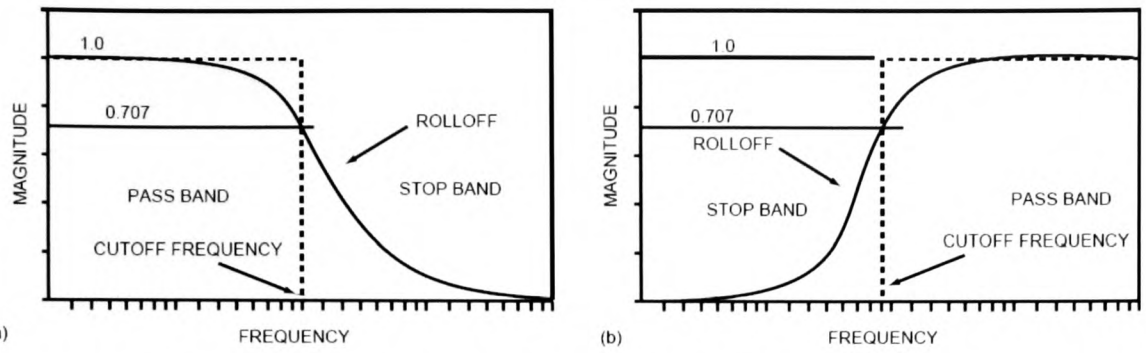


Figure 1.3: Frequency responses of simple low-pass (a) and high-pass (b) filters with their key attributes.
A *pass band* denotes the signal frequencies that are allowed to pass through the filter, a *stop band* denotes the signal frequencies that are attenuated by the filter, the *cut-off* frequency separates the pass-band from the stop-band and *roll-off* indicates the rate of transition between pass-band and stop-band. Excerpt from (Whitaker 2005) p.340.

In order to obtain frequency information in DWT the image is processed via a combination of low-pass and high-pass filters recursively (Subband Coding). In the context of Wavelet Transform this combination of filters is also called a *filter-bank* or simply *wavelets*.

Figure 1.3 depicts the frequency responses of two examples of generic high- and low-pass filters. The frequency response is a visual representation of how the filter affects different frequencies of an input signal. The magnitude corresponds to the amount of the signal that is allowed to pass through at the respective frequency. Higher magnitude means that a greater portion of the signal is allowed to pass through and lower magnitude means that the signal is attenuated by the filter.

In order to apply a filter in discrete real-space samples such as a digital image, the sample data (in the case of DWT the pixels) are convoluted with the filter coefficients. The operation is called the *convolution summation* (Smith 1999) and for discrete signals it is generally defined as:

$$y[i] = \sum_{j=0}^{M-1} h[j]x[i-j] \quad i=0, 1, \ldots, n \qquad \textbf{[eq 1-1]}$$

where $x[]$ is an N point signal running from 0 to N-1, $h[]$ is an M point signal running from 0 to M-1, $y[i]$ is the output of the convolution summation and i determines the

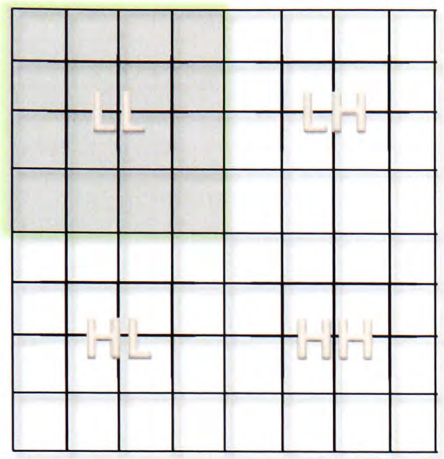


Figure 1.4: Layout of the coarse (LL), horizontal detail (LH), vertical detail (HL) and diagonal detail (HH) frequency subbands after one level of Discrete Wavelet Decomposition.

output sample that is being calculated. In the case of DWT the roles of the two convolved signals are taken by the pixel data and the filter coefficients. Note that since the filter has fewer coefficients than the number of pixels in any given direction, the coefficients are repeated during the convolution process cyclically.

Applying the wavelet transform to an image results in a low resolution image and a series of detail images (subbands). The low resolution image (denoted by LL in Figure 1.4) is obtained by iterative blurring of the original, whereas the detail images (denoted by LH, HL and HH in Figure 1.4) are the information that is lost during the operation. The DWT for the n^{th} layer of decomposition is calculated by applying a separable filter bank, as follows:

$$LL_n(x,y) = \left[H * \left[H * LL_{n-1}\right] \downarrow 2,1\right]_{\downarrow 1,2}(x,y) \qquad \textbf{[eq 1-2]}$$

$$LH_n(x,y) = \left[H * \left[G * LL_{n-1}\right] \downarrow 2,1\right]_{\downarrow 1,2}(x,y) \qquad \textbf{[eq 1-3]}$$

$$HL_n(x,y) = \left[G * \left[H * LL_{n-1}\right] \downarrow 2,1\right]_{\downarrow 1,2}(x,y) \qquad \textbf{[eq 1-4]}$$

$$HH_n(x,y) = \left[G * \left[G * LL_{n-1}\right] \downarrow 2,1\right]_{\downarrow 1,2}(x,y) \qquad \textbf{[eq 1-5]}$$

where * is the convolution operator, $\downarrow$ 2,1 and $\downarrow$ 1,2 are the sub-sampling along the rows and columns respectively, LL_0 is the original image, H and G are low and band-pass filters respectively (hence the subband names LL, LH, LH and HH) and x,y the coordinates of the target coefficient. LL_n is obtained by low pass filtering resulting in a low resolution image. The detail images (LH_n, HL_n *and* HH_n) are obtained via band-pass filtering. The original image is therefore represented by a set of sub-images (subbands) at several scales, in other words a multi-scale representation.

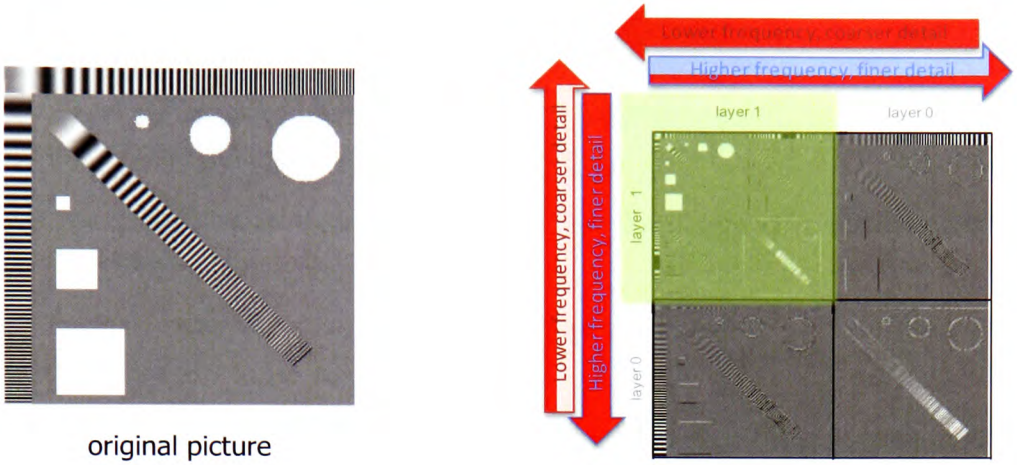


Figure 1.5: A two-level Discrete Wavelet Transform decomposition.

Images that have been decomposed using the wavelet transform contain subbands or sub-images that convey information regarding the directional details at a given resolution. For quadrature decomposition, the directional information includes horizontal (LH in Figure 1.4), vertical (HL in Figure 1.4) and diagonal (HH in Figure 1.4) details. Higher frequency subbands depict finer details or, in other words, information about sharper edges in the picture. Lower frequency subbands have information about coarser details or smoother areas of the picture (see figure 1.5 for an example). Sub-sampling of data at each recursion step ensures that the total number of generated DWT coefficients equals the number of the pixels of the original image resulting in a very compact representation. In wavelets domain high compression ratios can be achieved by discarding high frequency coefficients whilst maintaining high visual image quality.

DWT has several advantages over DCT (Mallat 1989; Antonini, Barlaud et al. 1992; Daubechies 1992; Mandal, Idris et al. 1999):

- Multi-resolution representation without additional storage requirements;
- Better adaptation to non-stationary signals;
- High de-correlation and energy compaction efficiency;
- Reduced blocking artefacts and mosquito noise;
- Better adaptation to the human visual system characteristics.

The major disadvantages of DWT over DCT are:

- Being part of the original JPEG image compression standard, DCT is at the heart of an enormous collection of compressed images proliferated by the WWW. It is therefore unsurprising that DCT has much wider commercial support as it is embedded in almost every software application requiring digital imaging functionality.
- DCT-based compression codecs are typically faster than DWT-based codecs.

Numerous applications of wavelet-based image retrieval algorithms exist in the literature, spanning diverse fields such as medical imaging, security, industrial materials defect detection, web-content filtering, GIS and OCR (Muller 2004; Qiao, Murtagh *et al.* 2004; Ghouti, Bouridane *et al.* 2006; Gupta, Lekshmi *et al.* 2006; Nabti, Ghouti *et al.* 2006; Chevallet, Lim *et al.* 2007; Dettori and Semler 2007; Gang and Zong-Min 2007; Han and Shi 2007; Lin 2007; Oliveira, Cirne *et al.* 2007; Scharcanski 2007; Selvan and Ramakrishnan 2007; Shih, Lee *et al.* 2007; Zhang and Cheng 2007; Zuyuan and Boesch 2007; Hiremath and Shivashankar 2008).

DWT is used in JPEG2000, an international standard for image compression and distribution (Santa-Cruz and Ebrahimi 2000; Santa-Cruz, Grospois *et al.* 2002). The format has been embraced by the image retrieval research community (Mandal and Liu 2004; Schaefer 2004; Lee 2005; Tabesh, Bilgin et al. 2005; Descampe, Vandergheynst et al. 2006; Teynor, Muller et al. 2006a; Au, Law et al. 2007).

Another popular wavelet-based compression scheme is Set Partitioning in Hierarchical Trees (SPIHT) (Said and Pearlman 1996b). SPIHT video and image codecs have been embraced by the research community due to its performance and easy implementation. Twelve years after the algorithm was first publicized, the interest is still strong with several new ideas that build upon and expand SPIHT in the research literature spanning fields such as efficient video coding (Martin, Lukac *et al.* 2006; Biswas, Frater *et al.* 2007; Yang and Cheng 2007), steganography (Chen 2008), medical imaging and spectrometry (Chen 2007; Wei, Zhao *et al.* 2007), hardware implementations (Nandi and Banakar 2008), and fingerprint recognition (Sung and Hsin 2007). The algorithm has been licensed by Silicon Imaging (SiliconImaging 2008) for commercial use in their digital video hardware and

software offerings and is part of the open source QccPack graphics library (Fowler 2008).

Since both algorithms are built around the wavelet transform they inherit all the advantages over DCT, providing excellent platforms for low-level feature extraction in compressed domain.

1.6 CBIR/CBIC design requirements and issues

In a review paper on image retrieval for the WWW (Kherfi, Ziou *et al.* 2004) the authors categorize the main issues of CBIR research from an architectural perspective resembling the schema in Section 1.4.1 (Figure 1.1a), *i.e.* query specification, image descriptors selection and estimation, indexing, similarity and matching, retrieval, refinement and performance evaluation. Two additional WWW-specific categories are identified: data gathering and web coverage. In another review paper on image retrieval (Datta, Joshi *et al.* 2008), the authors start from a wider perspective dealing with broader concepts such as: core technologies (*e.g.* features and signatures, similarity measures, classification and search paradigms), applications, real world (*e.g.* user intent, data scope and results visualization), evaluation.

The fundamental question that this work is attempting to answer is how to improve content-based image retrieval systems. Since this problem is extremely broad spanning several research domains, the focus of this work is narrowing onto a few specific issues, namely: to investigate the potential texture analysis in wavelets domain, to achieve feature extraction in native compressed domain and to improve the speed of the retrieval process. Performance refers to both the accuracy of the feature extraction algorithms and the speed of a query. This section is, thus, primarily aiming to pin-point the design issues that affect accuracy and speed in CBIR systems. In order to implement an experimental platform for empirical validation of this work, design decisions had to be made with regards to the query definition and the evaluation of the system. The issues related to those two subjects are also discussed in order to justify the respective design choices. Therefore, at the intersection of the taxonomies mentioned in the review papers above and the scope

of this thesis, this section gives a high level view of the issues related to three adjacent subjects: Query specification and scope, Performance and Evaluation.

1.6.1 Query specification and data scope

Query specification is a general issue of information retrieval involving the translation of the user needs not only into a form that is understandable by the machine but also into a form which accurately represents those needs (Pao and Lee 1989). Query specification is hindered by the ability of both the user to accurately express his requirements and of the retrieval system to provide a versatile and rich interface for mapping those requirements into machine-readable format.

The query specification options offered via the CBIR system interface are linked to the data scope of a specific application. Narrower data scope allows for more straightforward interfacing (Smeulders, Worring *et al.* 2000). For example, in face recognition the system input will almost exclusively be an image containing the unknown face. Similar conditions apply for fingerprint or corporate logo recognition. This type of query, also known as query by example (QBE) is very popular for content-based image retrieval (Chang and Kuo 1993) in narrow data scopes or image domains where the global image features are of interest.

User interfaces also depend on what features does the search engine employ, for example, low-/mid-/high- level features, text keywords. It is therefore important for the UI to manifest the available underlying functionality to the user in an efficient way, for example, for a colour-based retrieval the user could be presented with a colour pallet browser interface or a colour-picker tool in order to select a colour shade from a sample image. To this end, a number of options for CBIR querying have been proposed, most of which have been introduced by the QBIC system (Flickner, Sawhney et al. 1995). The QBIC interface allows users to perform colour-based queries using sliders or palettes, in order to specify the desired colour. Similarly, palettes with samples are used for the selection of texture. Finally, the user is provided with the option to draw a rough sketch of the desired shape (Flickner, Sawhney et al. 1995). Since the latter relies on the user's artistic expression ability other alternatives to query-by-sketch (Pentland, Picard *et al.* 1994; Smith 1997)

provided a set of primitive shapes which can be picked and combined in order to produce the query shape.

For generic image retrieval where the user intention is completely unknown and the image domain is very broad QBE and low-level features have not achieved enough accuracy for practical applications. To improve the accuracy of generic CBIR some complementary techniques to low-level CBIR have been proposed in the research literature such as relevance feedback and high-level semantics annotation. In relevance feedback the query specification is enriched by allowing the user to select the best (or worst) matches from the retrieved images over several iterations in order for the system to provide more accurate results. In high-level semantics annotation, the query specification can include textual keywords (manually entered or selected from a predefined list) in order for the system to narrow down the semantic space of the query. More details on both relevance feedback and semantic annotation are given in Section 1.6.2.

In this work the focus is on texture feature extraction. Since the image domain for texture-only images is narrow, QBE is used for the experiments following the paradigm of many recent studies in the CBIR research literature such as (Tsai, McGarry *et al.* 2006; Barcelos, Ferreira *et al.* 2007; Zhang and Zhang 2007; Avci 2008; Sengur 2008).

1.6.2 Performance: Accuracy

Sufficient accuracy of texture analysis techniques for practical applications is still an open issue in CBIR research (Deselaers, Keysers *et al.* 2008) whereas the interest in texture-centric CBIR systems such as Computer Aided Diagnosis is increasing (Megalooikonomou, Zhang *et al.* 2007).

From information retrieval principles (Pao and Lee 1989) the fundamental issues related to the feature extraction and signature formation processes are: a) to ensure that the specific signature accurately represents the specific feature and invariant in any other way and b) to ensure that the process has the appropriate level of discrimination ability between visually dissimilar states of the feature. In other words, (a) is the ability of the feature analysis algorithm to characterize the respective low-level feature in numerical format so that the value of the numerical representation is

a function of the visual variations of the specific low-level feature only. (*i.e.* variations in texture shouldn't affect the colour feature signature of an image and vice versa). Furthermore (b) means that the feature extraction and signature calculation processes attain adequate discrimination ability so that visual variations of a feature are reflected to numerical variations in the signature. Ideally, this variability of the signature should be high enough in order to discriminate between visually dissimilar low-level features but also low enough in order to detect visually similar features. The latter is a very difficult problem to solve unless there are distinctive classes of images involved.

In this work the focus is on texture analysis in wavelets domain. An extensive survey of the current status quo in the specific area is provided in Chapter 2 along with a detailed critical analysis of the shortcomings of the existing state-of-the-art and the unexplored opportunities that form the motivation for this research.

In one of the most cited reviews in CBIR research (Smeulders, Worring *et al.* 2000) the problem of accurately describing image content via digital processing is split into two major perceptual "gaps": the *sensory gap* and the *semantic gap.* Recent review papers such as (Kherfi, Ziou *et al.* 2004; Liu, Zhang *et al.* 2007; Datta, Joshi *et al.* 2008) confirm that the two "gaps" still reflect the best global taxonomy of CBIR accuracy issues. Below, some of the major issues that plague CBIR accuracy are identified, using the above categorization as a starting point. The focus is on those issues that formed the motivation for this work.

1.6.2.1 The sensory gap

In (Smeulders, Worring et al. 2000) the *sensory gap* is defined as "[...] *the gap between the object in the world and the information in a (computational) description derived from a recording of that scene.*" This involves several variables during the shooting of a photograph, for example, lighting conditions and camera position, which affect the visual representation of an image without affecting the semantic content (a zoomed out rotated image of a bicycle still shows a bicycle!). Studies of the human visual system (HVS) have shown that it decomposes a perceived image into various frequencies and orientations but also that it is very capable at distinguishing shapes and texture even between images with large variations in

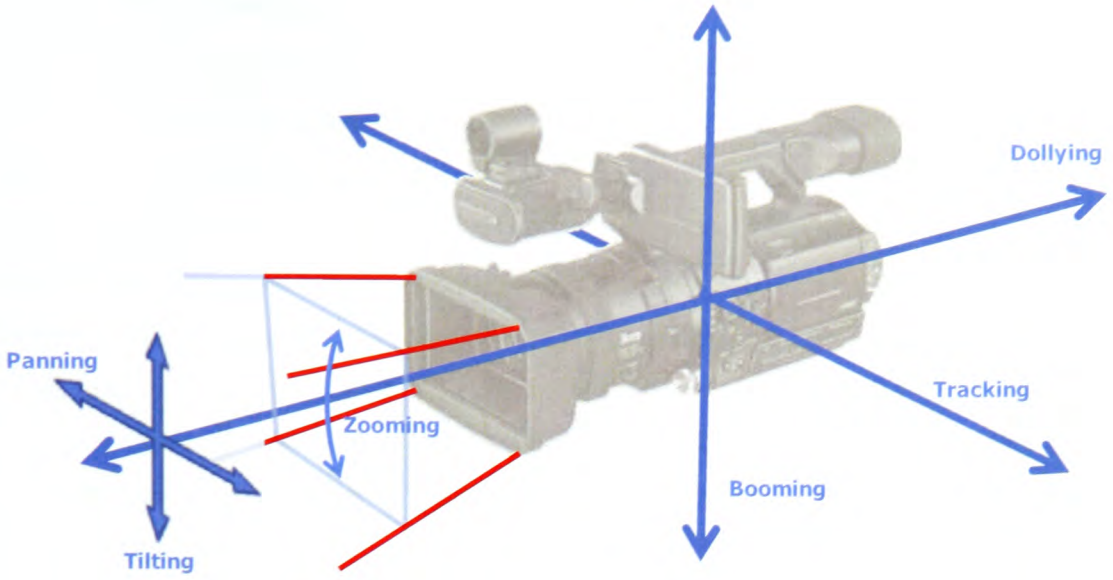


Figure 1.6: Basic Camera Operations.
Based on a concept by (Mandal, Idris *et al.* 1999)

illumination (Campbell and Robson 1968; Tuceryan and Jain 1998; Smith 2002; Zhang and Tan 2002; Neumann and Gegenfurtner 2006). From a digital imaging perspective the *sensory gap* of traditional texture analysis techniques is due to low tolerance to camera operations (see Figure 1.6) and lighting inconsistencies (Levine 1985).

Several low-level feature algorithms that remain "invariant" to distortions such as scale, rotation and tilt have appeared in the research literature (Porter and Canagarajah 1997; Ojala, Pietikainen et al. 2002b; Jafari-Khouzani and Soltanian-Zadeh 2005b; Cheng, Law et al. 2007; Kokare, Biswas et al. 2007; Pan, Bui et al. 2008). Specifically for rotation invariance, although several wavelets-based techniques exist in the CBIR bibliography, a pixel-domain technique (Ojala, Pietikainen et al. 2002b) is still one of the state-of-the-art algorithms when it comes to performance and low complexity. Similar or higher accuracy wavelet-based techniques have higher processing requirements and often use wavelet techniques that are incompatible with compression algorithms, thereby losing one of the advantages of operating in transform domain. Therefore, wavelet-based rotation-invariant methods have yet to achieve the performance balance (accuracy versus complexity) of pixel domain counterparts. Rotation invariance in texture analysis is analyzed in Chapter 2.

1.6.2.2 The semantic gap

On the other end of the spectrum, Smeulders *et al.* defines the *semantic gap* as "[...] *the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation.*" In (Liu, Zhang *et al.* 2007) the statement was rephrased in simpler form as "[...] how do we relate low-level image features to high-level semantics?". The authors identify five main approaches to reducing the 'semantic gap' these are:

a) Using object ontology to define high-level concepts. That is, to use a predefined vocabulary to map specific algorithmically derived object features to words. *i.e.* define lexically the colour, luminance, position, size and shape of specific areas of an image. For example, "there is a bright light-blue large area at the top left of the image" (Stanchev, Green Jr. *et al.* 2003).

b) Using supervised or unsupervised machine learning tools to associate low-level features with query concepts. In supervised learning the aim is to predict the semantic category label of an image out of a given set of labels whereas in unsupervised learning data clustering is performed automatically without prior knowledge. Examples of machine learning applications can be found in (Wang, Li et al. 2001) for unsupervised and (Minka and Picard 1997) for supervised.

c) Relevance feedback algorithms which request the user to rank (Cheng, Chien *et al.* 2008) or simply identify the most similar (or most irrelevant) images from the results and re-query. The process re-iterates until the user is satisfied by the results. Relevance Feedback was initially developed for textual retrieval (Salton 1989) and moved to CBIR in the early 1990s (Kurita and Kato 1993). Several approaches and use cases of relevance feedback in CBIR have appeared since (Zhou and Huang 2003; Zhi-Hua, Ke-Jia *et al.* 2006; Giorgio 2007; Saha, Das *et al.* 2007; Cheng, Chien *et al.* 2008)

d) Generating semantic templates which are low-level multi-feature signatures generated using a set of images selected to highlight the most prominent visual characteristics of a given semantic entity (Chang and Wang 1999). The idea is to train the system with those images so that a weighted low-level feature vector or Semantic Template (ST) encompasses all the characteristics of the specific semantic entity. The STs are then used in order to classify an image.

e) Making use of both the visual content of images and the contextual information collected from a variety of sources on the WWW including the image URL (Uniform Resource Locator), HTML tags (HyperText Markup Language), hyperlinks and surrounding text (Feng, Shi *et al.* 2004).

and combinations of the above (Mezaris, kompatsiaris *et al.* 2003; Vasconcelos 2007).

Although Semantic Annotation has received a lot of attention, the inherent issues of pre-processing, storage requirements, annotation and query specification accuracy still exist. Introduced by some online services such as flickr (Flickr 2008) and del.icio.us (delicious 2008) the idea of *Folksonomies* is gaining attraction in the field of semantic annotation. The concept behind *Folksonomies* is to give to the users the freedom to either assign one or more keyword *tags* to an image or to choose from a list of tags that have been already assigned by other users. This creates a taxonomy of an otherwise amorphous collection of information which can be used in an inverted format for searching. An interesting spin-off of this concept is the approach taken by Google (Google 2008c) recently, which is to request users to voluntarily annotate random images extracted from Google's collective image index repository. *Folksonomies* are attractive because currently, as it is further discussed in Chapter 2, the WWW can almost exclusively be searched for multimedia content using textual keywords.

1.6.3 Performance: Speed

Retrieval speed depends on the database size, the descriptors size and format, the indexing method, the similarity calculation and the latency/access overhead of the storage medium (Kherfi, Ziou *et al.* 2004). The similarity calculation accounts for most of the processing time during the querying process and its selection also affects accuracy (Smith 1997). The size and format of the descriptors are dictated by the feature extraction method and therefore they are specified at design time as a trade of between size and accuracy. The overhead related to the storage medium is outside the scope of image processing research. Therefore, for speed improvement the focus is on the indexing method and the similarity calculation.

1.6.3.1 Indexing methods & similarity calculation

Early attempts for efficient image indexing (Nasser, Meral *et al.* 1994; Brown and Gruenwald 1998) were plagued by curse of dimensionality. That is, the performance of the indexing scheme degrades as the dimensionality of the feature space increases. The dimensionality of image features is usually high ranging from tens to several hundred (Smeulders, Worring *et al.* 2000). In a study by (Weber, Schek *et al.* 1998) it was shown that, when the number of feature dimensions exceeds 10 sequential search becomes more efficient than any indexing method. Some indexing algorithms which have addressed the issue of high-dimensionality are (Van de Wouwer, Scheunders *et al.* 1999b; Corral, D'Ermiliis *et al.* 2005; Yu, Cui *et al.* 2007). An additional problem that was highlighted by (Amsaleg and Gros 2001) is that more indexing techniques are built for matching whereas image retrieval seeks similarity. In other words, the indexing system is not searching for a unique point but for degrees of similarity (or dissimilarity) in the feature space.

L_1- (Manhattan) and L_2- (Euclidian) distance are the most commonly used metrics for (dis)similarity calculation in image retrieval applications (Zhong and Defee 2007). Very few efforts have been presented intended to improve the efficiency of the distance calculation process to enable faster image retrieval. Two of the most prominent works are from (Berman and Shapiro 1999; Wilson and Bayoumi 2003). Efficient distance calculation is discussed in more detail in Chapter 2.

1.6.3.2 Working with compressed images

The discussion up to this point regarded systems for which the calculation and indexing of the feature vectors is performed *a priori* and is fully separated from the querying process. This type of processing has a number of disadvantages such as the need for storage space dedicated for the feature signatures, the need for the entire image database to be available for indexing *a priori*, and the inability to easily modify or update the feature. To overcome these shortcomings, a number of methods that jointly address the issues of compression and image indexing exist in the research literature.

One of the approaches that exist in the literature is to combine image compression and indexing as two parallel processes of the same system (Vassilakopoulos,

Manolopoulos *et al.* 1993; Liang and Kuo 1999; Fahmy, Black *et al.* 2006). The use of combined systems requires re-processing and re-compression of the entire database and they are not compatible with systems employing one of the existing compression standards. An alternative approach to combined compression and indexing is to compress images in such a way that it facilitates feature extraction (Chang and Carin 2006). That is, the coefficients that are most important for calculating the respective feature signatures appear earlier in the bit stream. This methodology suffers from the same problems as the combined compression and indexing albeit without the requirement for extra storage space for the signatures. In both cases full reprocessing is required if the extracted features are modified or amended.

The issues of both aforementioned methods are addressed by CBIR systems which extract the features directly from pre-compressed images. This provides the flexibility to work on dynamic image databases and also to select, modify or amend the extracted features per query specification. The latter is an important advantage considering that feature selection is usually specified on desirability and per-query requirements basis (Kherfi, Ziou *et al.* 2004; Datta, Joshi *et al.* 2008). Furthermore, no permanent storage space is required as the indexing keys are not pre-calculated. On-the-fly indexing is very important for applications such as remote sensing where resources such as local storage and network bandwidth are limited and content-based classification of the imagery has to be performed prior to full transmission (Chang and Carin 2006). For CBIR systems in compressed domain retrieval, speed is affected by the feature extraction method and by the level of decompression required for extraction of the necessary image data (See Section 1.5), in addition to the factors mentioned at the beginning of Section 1.6.3. By inspection of Figure 1.2 it becomes apparent that there are significant processing savings in being able to extract feature information close to (or ideally directly from) the bit stream (point E). From the perspective of the popular wavelet-based compression algorithms, existing research for retrieval from compressed images has almost exclusively focused on JPEG2000 whereas no such technique has been developed for SPIHT. The respective works are described in detail in Chapter 2.

1.6.3.3 Retrieval speed in the research literature

In a study of the medical applications for CBIR (Muller 2004) the author opines that the speed of a query is a very important characteristic especially for interactive systems and CBIR systems should be evaluated against it. Similarly, Nabti in (Nabti, Ghouti *et al.* 2006) underlines the importance of performance in terms of both speed and accuracy in security applications such as iris recognition. In (Kherfi, Ziou *et al.* 2004) the authors discuss image retrieval in the WWW and highlight that, as the size of the web increases and its structure becomes more and more complicated, faster indexing algorithms are required to enable more efficient traversing techniques.

Several studies on CBIR such as (Fahmy, Black *et al.* 2006; Lew, Sebe *et al.* 2006; Liu, Zhang *et al.* 2007; Datta, Joshi *et al.* 2008) highlight the fact that, despite the increase in mass-storage density, processor and data communication transmission speed, the demand for performance continues to surpass the capabilities. Several reasons for that are pin pointed such as the ever increasing size scale of systems whilst requiring real-time access and processing, the bandwidth and processing constraints of the increasingly popular mobile devices and the inability of current database systems to efficiently performing operations on non-fixed size elements like images and feature signatures. Concluding, retrieval speed has been largely overshadowed in the image retrieval research by the efforts to improve the accuracy of the feature extraction methods.

1.6.4 Evaluation of CBIR/CBIC systems

Evaluating the accuracy of CBIR systems in semantically rich environments such as generic image retrieval is difficult due to the nature of human visual perception (Niblack, Barber *et al.* 1993). More specifically this is due to the inconsistency between the perceptions of different people regarding the same image. Thus the problem of accurately evaluating a generic image retrieval system lies on that it is impossible to establish a ground truth for image similarity (Smeulders, Worring *et al.* 2000). A common approach for the evaluation of generic image retrieval systems is to ask a group of impartial users to assess the performance of the system by performing a number of test runs (Fahmy, Black et al. 2006; Teynor, Muller et al. 2006a).

This ambiguity is not true for systems with *a priori* established and labelled image categories or training sets which can be used as the ground truth. The problem in those cases lies on accurately representing the image category by the appropriate feature signature. In such cases where the image domain is narrow the most commonly used metrics for evaluating the systems are *precision* and *recall* (Smith 2001; Huang, Dai et al. 2006; Vasconcelos 2007; Datta, Joshi et al. 2008). These metrics are defined as follows:

- *Precision* is the ratio of relevant images retrieved over the total of retrieved images,
- *Recall* is the ratio of relevant images retrieved over the total of relevant images in the database.

Typically, a combination of the two metrics is used in the form of a precision/recall curve.

Further to *precision* and *recall*, another metric is *fall-out* which assesses the level of relevant but not retrieved images by measuring the efficiency of rejecting non-relevant items. *Fall-out* is defined as the ration of irrelevant retrieved images over the total of irrelevant images in the database.

Another popular metric is the Average Precision (AP). For a given query, a ranked list of the retrieved images is generated. Precision is then calculated at all levels of retrieved images and AP is derived by averaging all the results. Thereby the relevant images which are ranked higher in the list carry more weight in the final result. Comprehensive reviews of evaluation metrics are given by (Huijmans and Sebe 2005) and (Smith 2001). In this work evaluation is performed using the AP metric due to its straightforward and easy implementation and its popularity in the literature. The popularity of the metric in the image retrieval literature also enables the comparison with existing works.

Evaluation of a CBIR system also requires a representative dataset. In generic image retrieval there have been several standardization attempts (Clough, Mueller *et al.* 2005; Benchathlon 2008) for a common platform of CBIR system evaluation encompassing image sets, test sets and metrics.

A popular data set used for benchmarking generic image retrieval is the Corel Stock Photos (Corel 2008) although several shortcomings of this approach have been discussed in (Muller, Marchand-Maillet *et al.* 2002). These shortcomings include the fact that the images are copyrighted, they are not free and reportedly include highly dissimilar image groups with relatively high within group similarity, thereby affecting positively the performance of the algorithms. In order to overcome those problems the retrieval experiments were performed on an in-house compiled database of approximately 1000 images with mean size 512×256 pixels available at (Voulgaris 2008) spanning a plethora of diverse content. The images were collected from a variety of on-line royalty-free sources focusing on the breadth of different themes and image structure (i.e. various levels of detail, textures, shapes).

In texture analysis where the scope is much narrower and ground truth can be established more easily, some of the most popular evaluation datasets are the VisTex (MIT 2002) database and the Outex (Ojala, Maenpaa et al. 2002) database. Representative use cases include (Bashar, Matsumoto *et al.* 2003; Liapis and Tziritas 2004; Kokare, Biswas *et al.* 2005). Another very popular data set for benchmarking image retrieval and classification systems is the texture images contained in the Brodatz photograph album (Brodatz 1966) with representative use cases (Jafari-Khouzani and Soltanian-Zadeh 2005b; Huang, Dai *et al.* 2006; Pi, Tong *et al.* 2006; Kokare, Biswas *et al.* 2007; Hiremath and Shivashankar 2008; Pan, Bui *et al.* 2008). The latter is adopted in this work. Specifically, the Brodatz image set that is publicly available via the Outex Image Database of the University of Oulu, Finland (Ojala, Maenpaa et al. 2002), is used as the basis to generate four subsets to address requirements of different experiments. Another set of the Outex database which includes texture images captured at several rotation angles is used in Chapter 4 where rotation invariance in texture characterisation is addressed. A major advantage and motivation for incorporating this set into this work is the fact that it was constructed under controlled illumination and spatial resolution conditions to ensure uniform imaging geometry.

1.7 Conclusions & research Aims

Below is a summary of the conclusions that were drawn through this chapter with references to the respective sections:

GENERAL CONCLUSIONS

Current state-of-the-art has not yet delivered efficient and accurate indexing of image databases (Section 1.1) as most real-life database systems still depend on text which is provably inefficient for indexing (Section 1.2). CBIR is based on low-level features and is suitable for automated computation (Section 1.3). CBIR is currently more applicable to narrow-image-domain applications (Section 1.3). Even for narrow-image-domain applications there are still sensory-gap issues that need to be resolved (Section 1.6.2). Generic CBIR which applies to broad-image-domains is an unsolved problem (Section 1.4) requiring bridging of the semantic-gap (Section 1.6.2) and finding proper ways of evaluating systems (Section 1.6.4).

FEATURE EXTRACTION IN COMPRESSED DOMAIN

Feature extraction in transform compressed domain is desirable because it provides a model which is similar to the HVS and most imagery is stored in compressed format from capture (Sections 1.5 & 1.6). The later is also an incentive for finding ways to extract features directly from the compressed bit-stream (Section 1.6.3). The need for on-the-fly feature extraction also exists for systems with limited storage and data transfer capabilities such as mobile devices and remote sensors (Section 1.6.3.2). SPIHT and JPEG2000 are two wavelet-based compression codecs with wide adoption by the research community and commercial implementations serving as excellent platforms for feature extraction in compressed domain (Section 1.5).

ACCURACY OF TEXTURE ANALYSIS

Sufficient accuracy of texture analysis techniques for practical applications is an open issue whilst the interest for texture retrieval applications is increasing (Section 1.6.2). There are still pixel domain techniques which offer better ratio of accuracy over processing for texture features than wavelet-based techniques (Section 1.6.2).

EFFICIENCY OF SIMILARITY CALCULATION

Speed improvement and evaluation is a largely uncharted territory in feature extraction accuracy in image retrieval research literature (Section 1.6.3) especially the optimization of similarity calculation (Section 1.6.3.1).

In a recent review (Lew, Sebe *et al.* 2006) of Content-based Multimedia Information Retrieval (MIR), the authors report that the common theme at panel discussions in leading conferences such as ACM's Multimedia Information Retrieval and Conference on Image and Video Retrieval was that there were no solved problems! According to the authors the central conclusion of most panels was that: "... [in all forms of multimedia retrieval] *the general problem remains largely unsolved. In summary, all of the general problems need significant further research.*"

The rationale for this research builds upon the overall conclusion that CBIR has not yet reached sufficient performance balance of accuracy, efficiency and speed for wide adoption in practical applications. Summarizing, attention was drawn on CBIR in transform domain and in particular wavelets because of the excellent characteristics for compression and texture extraction applications and the wide adoption in the research community and industry. Accuracy of texture analysis and in particular the sensory gap has yet to be resolved. Considering the demand for wavelets-based techniques it would be useful to match or exceed the performance balance of similar pixel-based techniques. Speed improvement remains an uncharted territory as is feature extraction directly from the bit-stream of compressed images.

Therefore, the general aim of this research is to provide the means for improving accuracy and speed of content-based image indexing, retrieval and classification in wavelets domain. In particular, this research sought to:

- Suggest new and build upon existing techniques for improving the accuracy of texture extraction in wavelets domain; ideally these should be compatible and capable of being extended to compressed domain.
- Suggest methods for improving the speed of CBIR/CBIC systems;
- Suggest methods for feature extraction directly from the compressed-bit stream with no decompression stage required.

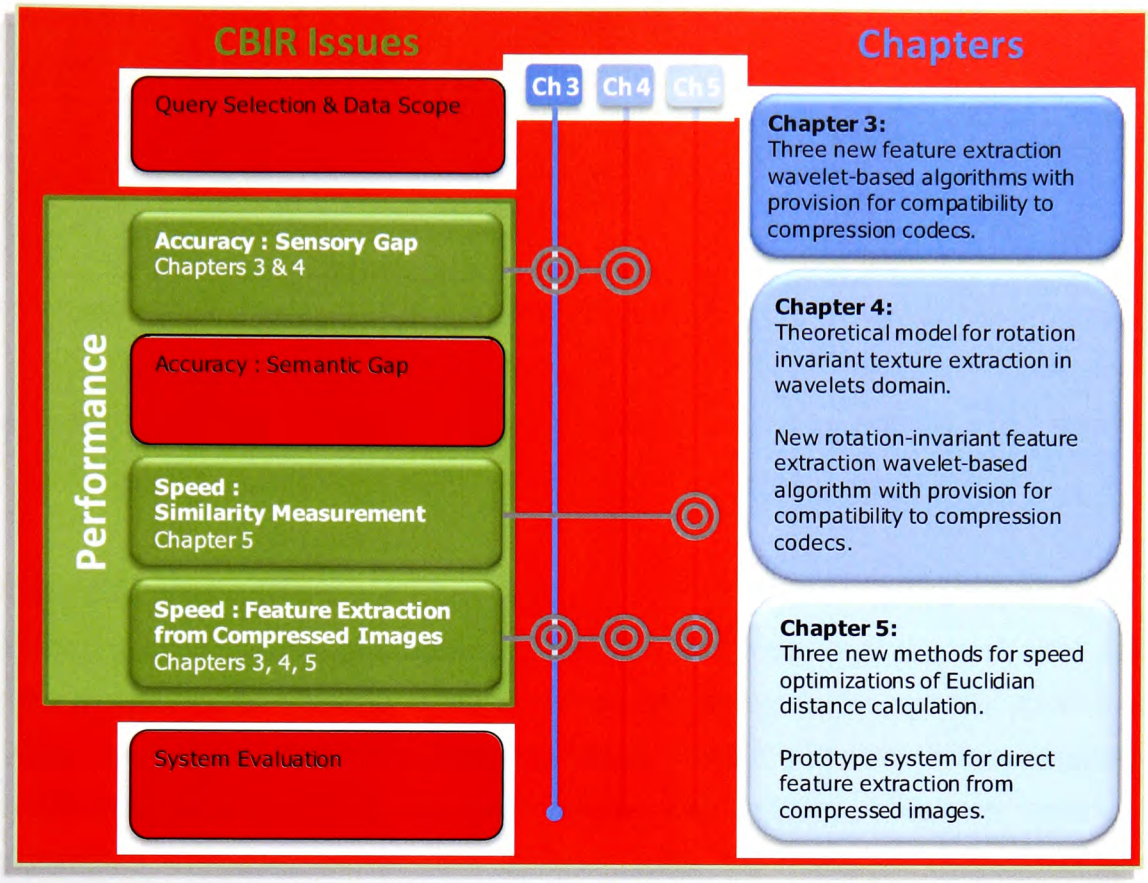


Figure 1.7: Positioning of contributions with respect to the CBIR research taxonomy discussed in Section 1.6.

Figure 1.7 depicts the positioning of the contributions presented in this thesis with respect to the current CBIR research practice based on the taxonomy of Section 1.6. Part of the work presented in this thesis has been published in national and international refereed conferences. The relevant citations are provided in the "Conclusions" sections of the respective chapters.

Chapter 2: Literature Review & Research Rationale

2.1 Abstract

This chapter offers a review of the research literature pertaining to current image content characterization techniques. All techniques are briefly described followed by a section summarizing their strengths and weaknesses. The chapter concludes by outlining the unresolved issues that formed the motivation for this work.

2.2 Introduction

Building upon the conclusions drawn in Chapter 1 and the generic research aims that were formed, here follows a study of the current status in the specific areas that define the scope of this thesis. More specifically, following the top-bottom approach that is depicted in Figure 2.1, the chapter begins with a survey of the most important commercial and experimental CBIR systems giving the reader an overview of the current offerings. Following is a detail account of texture in the context of image processing covering applications and texture analysis techniques. The literature survey concludes with a comprehensive analysis of the state-of-the-art in the areas pertaining this work *i.e.*, texture analysis in wavelets domain, texture extraction from compressed images, rotation-invariant texture extraction and optimization of similarity calculation. The chapter concludes with a critical analysis of the findings highlighting the unresolved issues that formed the motivation and the scope for this work.

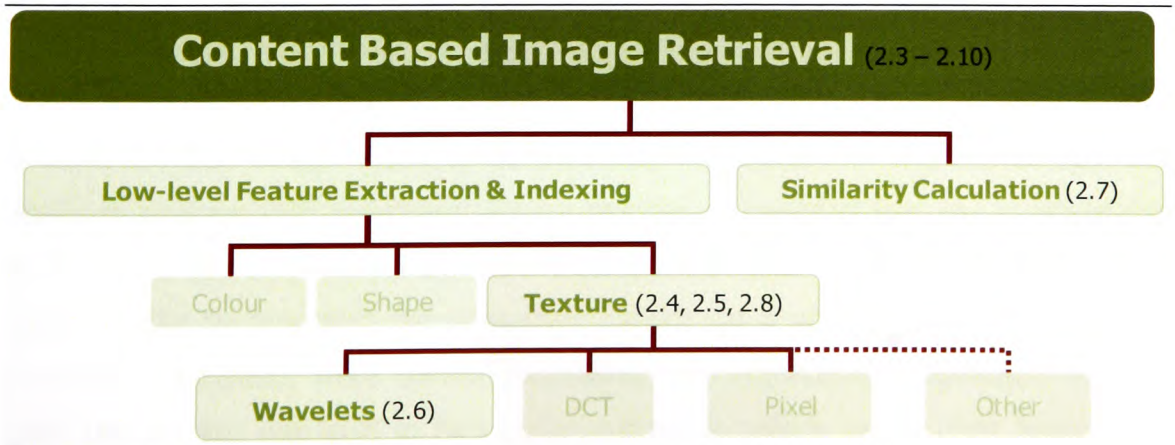


Figure 2.1: Overview of subject coverage in the literature survey.

For further information on CBIR the reader is pointed to the several excellent review papers on the subject. Comprehensive review papers which offer a 360 degrees view of the early CBIR research landscape are (Mandal, Idris *et al.* 1999; Smeulders, Worring *et al.* 2000) and recently (Lew, Sebe *et al.* 2006; Datta, Joshi *et al.* 2008). Equally comprehensive reviews with a bias towards retrieval from the World Wide Web are (Kherfi, Ziou *et al.* 2004; Spink and Jansen 2006; Tjondronegoro and Spink 2008). Content-based image retrieval in medical applications is studied by (Wang 2001; Muller 2004) and (Castellano, Bonilha *et al.* 2004) which focuses on texture analysis. A review of CBIR with high-level semantics is given by (Liu, Zhang *et al.* 2007) and specifically on relevance feedback by (Zhou and Huang 2003).

2.3 Commercial & experimental systems

2.3.1 Commercial landscape

Very few commercial offerings incorporating CBIR/CBIC technology exist. Here the five most representative commercial solutions are discussed.

Developed by IBM in 1995, **QBIC** (Flickner, Sawhney et al. 1995) offers the ability to search by any combination of colour, texture, shape and textual keywords. Queries can be made using QBE, manual sketching of the desired shape, or by selection of colours from a specified palette. The most similar images are retrieved displayed in the form of thumbnails. QBIC is currently available as part of the "Image Extenders" plug-in modules for IBM's database product DB2 (IBM 2008). The only announced deployment of the QBIC system is the digital archive of the State Hermitage Museum in St. Petersburg, Russia .

Originally released in 1996 by Convera (Convera 2008), Visual **RetrievalWare**, was a content-based image indexing Software Development Kit. The toolkit had the ability to perform QBE searching using a number of low-level features in adjustable proportions. The user could vary the extent to which each of the features affects the query results by adjusting the respective weight value which ranges from 0 to 5. Convera's customers were almost exclusively law enforcement agencies. In April 2007 the product was sold to Norwegian enterprise search vendor Fast Search and Trasfer (FastSearch 2008) and in April 2008 Fast S&T became a subsidiary of

Microsoft. The RetrievalWare product name has been dropped and although Fast S&T includes a multimedia component in its enterprise search solution it is unclear whether it incorporates RetrievalWare or other CBIR technology.

Founded in 1999 by veteran scientists from the MIT MediaLab, Oxford University and INRIA, LookThatUp (LTU) is a company dedicated to image search and recognition solutions (LTU 2008). Based in Paris, France, LTU's main offering is the **Image Seeker** content-based visual search and retrieval product. Image Seeker offers both content-based and keyword-based search including the ability to upload an image for query-by-example searches. The software is advertised as capable of finding duplicate, clone or similar images using low-level features, image segmentation and unsupervised classification. The technology has also been repackaged under the name Image Filter for content-based image classification and filtering. LTU's solutions are targeting verticals such as law enforcement, trademark search and content filtering. Both Image Seeker and Image Filter have wide commercial acceptance with clients including: the FBI, the French Patent Office, the French and Italian Law enforcement agencies, Yahoo! Europe, Meredith Corporation and Corbis media. In 2005 the company was acquired by the Japanese software vendor Jastec Co. LTU is the only company which still has CBIR as its core product.

Based on the Photobook (Pentland, Picard *et al.* 1994) image retrieval research prototype, **FaceID** was a face recognition solution commercialized by Viisage Technology and deployed by US Police authorities. Photobook comprises a number of tools for browsing and searching images and image sequences based on the idea of semantics-preserving image compression. In contrary to typical image compression algorithms, semantics-preserving image compression is intended not to achieve high compression rates but to ensure that all essential content information is retained. In mid-2006, the merger between Viisage Technologym Inc. and Identix Incorporated gave birth to L-1 Identity Solutions (L1id 2008). The company now has a wide portfolio of products spanning biometrics (fingerprint, face ID and iris ID), secure credentialing and document authentication.

Founded in 1994, Virage (Virage 2008) offered the **VIR Image Engine**, an extensible software library for image and video feature extraction (Bach, Fuller *et al.*

1996). The engine incorporated a set of low-level features including global and local colour, texture and shape. Virage called these *universal primitives*. The modular architecture of the system allowed the creation of new plug-in primitives, customizable combinations of existing primitives, customized metrics and application-specific APIs. The product was offered as a dynamic or static library available for several OS platforms. Focusing on modularity and customization Virage targeted verticals such as medical imaging, CAD and trademark search. The company got acquired by Autonomy in mid-2003 hence becoming part of its enterprise search solutions portfolio. Autonomy is a data mining and enterprise search solutions giant, founded in 1996 as a Cambridge University spin-off (Autonomy 2008).

Concluding, the scarcity of available commercial offerings indicates the immaturity of the field to sustain practical applications. Although the offerings advertise several capabilities and features the deployments focus on very narrow semantic spaces such as biometrics and trademark search and/or include keyword annotation in order to improve relevance. The mergers and the acquisitions however indicate that the interest for CBIR is keen in the technology industry. Clearly there is still room for improvement before CBIR becomes commodity. Interestingly all offerings are start-ups founded during the 1990s and based on early CBIR technology which, given the numerous issues that have been highlighted in the literature, places doubt on how they compare against the state-of-the-art.

2.3.2 Image searching and the WWW

Considering the explosion of available multimedia content on the web, one would expect that many highly-sophisticated tools for accurate image retrieval are available. Looking into five of the most popular web search engines namely,

- Google (Google 2008a);
- MSN Live Search (Microsoft 2008);
- Lycos (Lycos 2008);
- Yahoo! (Yahoo! 2008);
- AltaVista (AltaVista 2008).

, the user can find dedicated search for images in all of them. A closer look reveals that in all cases search is restricted to textual keywords. In order to provide more accurate results, MS Live Search, Google, Yahoo! and AltaVista offer search

customization with limited flexibility. The customization options include: choice between pre-defined image sizes (*e.g.* small, medium, large), choice between B&W or colour only images, image context (*e.g.* news related, movies related) and file type. Google and MSN Live Search are taking customized search one step further, offering the option to filter the results which contain human faces, reportedly based on CBIR techniques. The query is still performed using textual keywords but the results are filtered in order to retrieve images containing faces with no further refinement or adjustment options with regards to the visual content.

In mid-2006, Microsoft announced in collaboration with the University of Washington (Snavely, Seitz *et al.* 2006), an innovative application of CBIR for image browsing called **Photosynth** (Photosynth 2008). Photosynth analyzes a large collection of photos from a specific theme and detects the relative 3-dimensional position and orientation of each photograph. This way the system can reconstruct a 3-D model of the scene using a mosaic of overlaid photos. The entire collection can then be browsed using the 3-D mosaic as a navigation GUI tool. A preliminary yet fully functional demo of PhotoSynth is currently on the project's website.

In a recent review paper, Tjondronegoro (Tjondronegoro and Spink 2008) offers a comparative review of 105 web-based search engines in terms of how they handle multimedia content. The search engines reviewed include both generic and application-specific solutions. The key findings of the report include that only 59 out of 105 search engines include support for multimedia retrieval, whilst, all but three of them rely exclusively on meta-data *i.e.* text-based. The three search engines that are mentioned to support content-based search are LTE's solution, the State Hermitage Museum in St. Petersburg and Yotofoto.com. However, the author of this thesis was unable to confirm Yotofoto's support for CBIR which leads to the conclusion that the functionality has been removed.

Another interesting finding of Tjondronegoro's report is that support for semantic annotation is also scarce, highlighting the difficulty of the problem. This is also backed by a survey by Deuel (Deuel 2004) reporting that most multimedia content on the WWW is not surrounded by enough text for inference. Yet given the ever

increasing growth rate of the level of interest in multimedia Web search (Spink and Jansen 2006; Tjondronegoro and Spink 2008) advances in the field are imminent.

2.3.3 Experimental Platforms

This section provides an overview of selected content-based image retrieval systems frequently cited in the research literature. Although citation traction offers an indication of the impact of a contribution in the research field, it favours older publications which have been available for a longer period of time. To compensate, two recent systems, CLAIRE and Anaktisi, are also included in this survey. For further information the reader is directed to any of the comprehensive review papers in the field of CBIR listed in Section 2.2.

Developed at UC Berkeley, **Blobworld** (Carson, Belongie *et al.* Aug 2002) is a CBIR that works directly on raw pixel data. The system segments an image into regions based on local colour and texture features. The so called "Blobworld" representation, which refers to the segmented version of an image, allows the user to get an insight into the way the system analyzes an image. In this way the user can inspect the cause for false results and adjust the query accordingly by selecting the appropriate regions of interest. Blobworld allows query at object level where "object" refers to a specific blob or region in an image. The authors report that this method provides more accurate results than using global image analysis. However, Blobworld is inflexible when the system determined regions (blobs) do not correspond to one or more of the actual regions of major interest.

Developed at Universite de Sherbrooke in Canada, **Atlas WISE** (Kherfi, Ziou *et al.* 2003) is a web-based image retrieval system. Atlas WISE traverses the web and collects both visual and textual data. Images are analyzed in order to extract features such as colour and edge histograms. Semantic keywords are collected from tags, captions and distilled nearby text content. The design focus of the system is on the relevance feedback mechanism used to improve the accuracy of the retrieval. The mechanism is unique in that it offers the capability to the user to designate both positive (desirable) and negative (undesirable) examples during the relevance feedback steps. The author reports that, compared to a benchmark implementation

of (Rui and Huang 2000), the hybrid positive and negative feedback approach offers better rate of accuracy improvement per iteration.

WEBSeek (Smith 1997) is a tool designed to perform content-based image retrieval on the web. The system utilizes textual categorization of the images found on the web and colour histograms in order to extract content feature information. The user first selects one or more of the extracted available categories and then chooses a query image from the given set. The search results can be refined using relevance feedback. WEBSeek builds upon the previous work on SaFe (Smith 1997) and VisualSEEk (Smith and Chang 1996). The general framework of SaFe provides the ability to search images based on the spatial arrangement of regions or objects. Regions and objects are defined at query time by the user in terms of spatial location, size, shape and colour. Safe can also perform queries based on spatial relationships, *i.e.* objects' positions relative to each other.

WebMARS (Ortega-Binderberger, Mehrotra *et al.* 2000) is a multimedia web-based search system that combines textual and visual keywords for HTML document retrieval. WebMARS retrieves the full HTML page instead of only the images. WebMARS is based on the MARS system (Flickner, Sawhney et al. 1995), which has been developed at the University of Illinois. The system characterizes images using a variety of features such as, colour, texture, shape and textual descriptors. The query can be performed either by choosing an example image or by selecting a particular colour or texture from the available palettes. Relevance feedback techniques on the query results are used for adjustment of the feature weights or even include different similarity measures (Rui, Huang *et al.* 1999).

SurfImage was developed by INRIA, France (Nastar 1998). The system offers a selection of low-level features, such as colour (histogram), shape (edge-orientation histogram) and texture (co-occurrence matrix). A number of more complex high level features can also be calculated such as eigen-images, deformable intensity surfaces and intensity surface curvature histogram. All available features can be combined in various user-defined ways. Finally, the system offers a relevance feedback mechanism which attempts to improve the accuracy of the results.

Netra means 'eye' in Sanskrit (Mandal, Panchanathan *et al.* 1997). In Netra, 2500 images from the Corel photograph collection are organized in 25 categories, each one containing 100 images. The images are segmented into regions of homogeneous colour and the following features are extracted: colour (code book quantization), texture (gabor wavelet transforms), shape and spatial location (curvature function, centroid, complex coordinate function). The user can click on one of the automatically segmented regions and then select one of the features to be used for querying. Alternatively, a query can be performed by specifying a colour from a palette and defining an area of interest.

In attempt to bridge the semantic gap, **CLAIRE** (Tsai, McGarry *et al.* 2006) is an image classification system which uses low-level features to automatically create semantic labels for image regions. The system performs a three-stage process. First an image is selected using QBE. Next, the image is tiled into fixed size regions and colour and texture information is extracted for each region. The low-level feature signatures are then mapped onto a set of predefined concept descriptors. That is, for each low-level feature a pre-defined list of semantic categories exist and the signature is classified to the respective category. Therefore, the two extracted features are represented by two semantic labels (*e.g.* colour: blue, texture: sky). Using Support Vector Machines (SVM), the two feature semantic labels are then combined in order to generate a semantic label for the region and then globally for the image.

Anaktisi (Chatzichristofis and Boutalis 2008b) means 'retrieval' in Greek. It is an image indexing and retrieval system developed at the Democritus University of Thrace, Greece. The system uses QBE and contains 31000 images from various collections. The user can select one of nine image collections, the feature extraction algorithm and pick any image from the collection as the query. The entire collection is sorted based on similarity to the query image. A selection of four feature extraction algorithms is offered which combine colour, texture and edge directivity information. The algorithms are based on fuzzy logic and pixel data for generation of the feature histograms and the differences are on the pair of features that they employ (permutations of colour, texture and edge directivity) and the size of the

generated descriptor (full and compact versions). An online demo of the system is available at (Chatzichristofis and Boutalis 2008a).

Concluding, although the experimental platforms presented in this section contain all the necessary functional subsystems of a CBIR (see Section 1.4) they primarily focus on one or two specific aspects of CBIR. These aspects include innovative query specification methods (Blobworld, WEBSeek, Netra), enhancements on the concept of relevance feedback (AtlasWISE, WEBSeek, SurfImage), and new content descriptors (CLAIRE, Anaktisi). Interestingly, there has been no successful attempt to build a system that would enhance and integrate the best of breed methods in order to offer a complete usable solution for generic CBIR. This can be considered as another indication of the immaturity of the field for mainstream adoption in usable applications which was highlighted in Section 2.3.1.

In the context of the original aims of this work, all presented systems would be greatly benefited by extracting feature information directly from compressed images for the reasons that were stated in Chapter 1. A more general conclusion is that none of the systems has found its way into a real-life application, thereby indicating that there is still room for improvement in terms of accuracy before the critical level which will enable practical use is reached.

2.4 Motivation behind the study of CBIR/CBIC systems and wavelets-based texture feature extraction.

This section provides the general rationale behind the decision to focus on content-based image retrieval and specifically texture analysis in wavelets-domain. This is achieved by providing answers to two fundamental questions: "*Why is there an interest in CBIR/CBIC systems?*", and "*Why is there an interest in wavelets-based texture feature extraction?*". This introductory discussion of this Section is further supported by an in-depth research survey which follows in the remaining sections of this chapter.

Why is there an interest in CBIR/CBIC systems?

Research interest in content-based image retrieval and classification has grown exponentially during the last decade, mainly due to the sudden explosion of available

processing power, and the resulting digital content storage and manipulation requirements. Existing application areas of CBIR/CBIC include: GIS, medical informatics, manufacturing, security and IT. The demand for applications is growing and so is the need for faster and more accurate CBIR/CBIC systems.

The upwards trend is also witnessed when focusing into the area of wavelet-based texture analysis and retrieval.

Why is there an interest in wavelets-based texture feature extraction?

Research on image content characterization from low level features dates back to the 1970s (Julesz 1975; Weszka and Deya 1976; Haralick 1979). Pixel-based techniques have thus reached a maturity stage in terms of retrieval and classification accuracy with some recent iterations still representing the state-of-the-art. During the 1990s the research interest has shifted towards transform-based methods because of their compact representation of the frequency response of an image which is convenient for low-level feature extraction.

For the last five years a consistent upward trend is witnessed for wavelet-based texture analysis within the broader research field of image retrieval. The similarity between an image representation in wavelets domain and the HVS has made the former an excellent choice for feature analysis, especially texture. Research interest in wavelets is also vivid in image compression due to their excellent visual quality preservation characteristics and the compact representation of the spatial-frequency characteristics of an image. Numerous excellent examples of wavelet-based techniques appear in the research literature.

2.5 Texture in image processing

2.5.1 What is texture?

Texture conveys the visual information that describes the nature, composition and three dimensional shape of an object (Tuceryan and Jain 1998). In two-dimensional image processing those features map to patterns with characteristic intensity variations, colour and size. Therefore, texture can be regarded as a similarity grouping within an image (Rosenfeld and Kak 1982).

From the Human Vision System (HVS) perspective, the pattern properties of texture are manifested to the perceptual signature of an image area in terms of lightness, contrast, uniformity, density, roughness, regularity, linearity, frequency, phase directionality, coarseness, randomness, fineness, smoothness, granulation *etc.*, of the texture as a whole (Levine 1985). Accordingly, it is this set of texture related information that is used for scene interpretation (Tuceryan and Jain 1993). In 1966 Brodatz *et al.* compiled an album (Brodatz 1966) of a very large collection of textures.

2.5.2 Texture in CBIR and CBIC.

Texture has been proven a particularly effective low-level feature for image characterisation in CBIR research. Most content-based image retrieval systems use texture as the only (in application-specific systems) or in conjunction with other low-level features for indexing and retrieval purposes (Bach, Fuller *et al.* 1996; Mandal, Idris *et al.* 1999; Smeulders, Worring *et al.* 2000; Castellano, Bonilha *et al.* 2004; Deselaers, Keysers *et al.* 2004; Datta, Joshi *et al.* 2008).

Although computational characterization of texture dates back to the 1970s (Julesz 1975; Haralick 1979), recent studies such as these by Deselaer (Deselaers, Keysers *et al.* 2008) and Datta (Datta, Joshi *et al.* 2008) indicate that the performance of texture analysis needs to be improved before practical CBIR applications become commoditized.

2.6 Overview of texture analysis techniques

According to the taxonomy presented in (Schalkoff 1989) and (Materka and Strzelecki 1998) texture analysis techniques can be categorized according to the mathematical tools that they use as: Structural, Statistical, Model-based and transform (spectral). Here follows an overview of the four approaches.

2.6.1 Structural techniques

The principle of *structural approaches* (Haralick 1979; Levine 1985) lies on the representation of texture by: a) well defined primitive elements (microtexture), and b) a specified order of spatial compositions (macrotexture) of those elements. To

describe a specific texture, the microtexture elements and the macrotexture placement rules need to be defined. A primitive is a connected set of cells characterized by grey level, shape or homogeneity. A function of position or the types of neighbouring primitives may define the rules for positioning a primitive element.

The structural approach provides an easy symbolic description of an image which is particularly suitable for synthesis tasks. The abstract descriptions are challenging when describing natural textures due to the inconsistency of micro- and macro-structural characteristics and the unclear differentiation between them. Nevertheless, several tools for structure texture analysis exist in the field of mathematical morphology (Serra 1982; Chen and Dougherty 1994). Structural approaches can be useful in circumstances where the uniformity of micro- and macro-texture features can be established. Such conditions exist in medical imaging applications (Megalooikonomou, Zhang *et al.* 2007).

2.6.2 Statistical techniques

Statistical approaches represent the texture via the non-deterministic properties of the distributions and relationships between the intensity levels of an image (Materka and Strzelecki 1998). Typical methods include the autocorrelation function, co-occurrence matrix and random field models. Statistical methods categorize textures as smooth, fine, coarse, granulated, rippled, regular or linear. It has been shown (Weszka and Deya 1976; Materka and Strzelecki 1998) that techniques based on second-order statistics have better discrimination capabilities than many techniques which are based on transform or structural approaches. In a study of texture statistical properties and the correlation with the human ability to discriminate textures (Julesz 1975), Julesz reports that when the second order moments of grey level texture images differ the discrimination is immediate. Equal 2nd order moments but different 3rd order moments are more demanding in terms of cognitive effort. This indicates that for automated texture-based image characterization, statistics of the 1st and 2nd order are important (Niemann 1981).

The co-occurrence matrix (Haralick 1979) is a widely adopted 2nd order statistical feature for texture characterization. Features based on the co-occurrence matrix

were demonstrated to potentially provide effective texture discrimination in biomedical images (Lerski, Straughan *et al.* 1993; Materka and Strzelecki 1998). Furthermore, the multidimensional occurrence and co-occurrence matrices have been proven to have excellent characteristics for texture analysis (Valkealahti and Oja 1998; Ojala, Pietikainen et al. 2002b).

2.6.3 Model-based techniques

Model-based methods use fractal and stochastic models in order to characterize the texture content of an image by using the generative image model and the stochastic model, respectively (Materka and Strzelecki 1998). The parameters of each of the two models must be estimated and then used for feature characterization. Typically, stochastic model parameters require complex computation. The fractal model is reportedly effective for describing of selected natural textures as well as texture characterization and classification (Pentland 1984; Chaudhuri and Sarkar 1995; Kaplan and Kuo 1995; Materka and Strzelecki 1998). However, the fractal model is unsuitable for characterizing local image structures and it lacks orientation selectivity.

Another form of model-based technique is the autoregressive model (AR) which analyses relationships between neighbouring pixels (Kashyap and Khotanzad 1986). Prominent use cases of the AR model are in pixel domain texture analysis (Tardif and Zaccarin 1997; Ojala, Pietikainen et al. 2002b; Cariou and Chehdi 2008), motion estimation (Heikkila and Pietikainen 2006) and Fourier transform texture analysis (Lillo, Motta *et al.* 2008).

2.6.4 Transform-based techniques

Transform-based techniques exploit the characteristics of the translation the pixel data of an image in a different domain (such as spatial-frequency) that is closely related to texture perceptual features. Transform-based techniques are closely related to compression. Since most imagery is stored in compressed form, the research interest during the last decade has been shifted towards transform domain texture analysis. Driven mostly by the widespread use of the JPEG image compression standard, a lot of work has been done on techniques based on the DCT domain (Chang 1995; Shneier and Abdel-Mottaleb 1996). As the advantages of

wavelets for both feature analysis and compression became apparent, an increasing number of researchers moved to the wavelet-based techniques, for example, (Lu 1997; Descampe, Vandergheynst *et al.* 2006). Other transformations which have been applied for feature extraction are the Fourier transform (Rosenfeld and Weszka 1980) and Gabor filters (Daugman 1985; Bovik, Clark *et al.* 1990; Porter and Canagarajah 1997).

2.6.4.1 Discrete Cosine Transform (DCT)

Although the scope of this work is focusing on wavelet domain techniques, a number of representative DCT-based techniques are also presented.

Chang (Chang 1995) has proposed a texture-based indexing technique using the DCT subband energy to define the texture features. The DCT/Mandala transform is used in order to derive N^2 bands for an $N \times N$ DCT transform. The feature signature is compacted through the Fisher discriminant technique. The feature elements in the transformed domain are mapped to a set of eigenvectors. The Mahalanobis distance is used as a metric for image similarity.

Shneier and Abdel-Mottaleb (Shneier and Abdel-Mottaleb 1996) suggested an image retrieval technique based on JPEG. Initially the algorithm selects a set of $2m$ windows which are then randomly matched provided that each window has only one partner. For each pair of windows, a bit is allocated in the m bit index. For each window the average of each DCT coefficient is computed giving a 64-dimensional feature vector. For each feature value and each window pair, the index is computed by comparison of the values of the first and second windows. The Euclidian distance is used as the similarity metric between their keys.

Jiang *et al.* (Jiang, Armstrong *et al.* 2002; Jiang, Armstrong *et al.* 2004) revisits the DCT equations by applying a Taylor series to expand the transform into a smaller number of items. The high order items and the higher frequency band coefficients are ignored. As a result, pixel intensity values for grey-scale images, can be extracted by simpler operations. This method is augmented in (Jiang, Armstrong *et al.* 2004) by incorporating all the colour channels of an image and Local Binary Partitioning for further improvement of texture discrimination. The method has low

computing cost since no full decompression is required. The approximated image reserves the content features, which are sufficient for indexing and browsing as evidenced by experimental results.

2.6.4.2 Wavelets

Due to the spatial-frequency nature of the wavelet transform, wavelet-based techniques are particularly effective for texture analysis (Chitre and Dhawan 1999). During the wavelet transformation process, the image is decomposed into a set of subbands (or frequency channels) with narrower bandwidths at lower frequencies. Due to its multi-scale property, the transform inherently separates the "smooth" areas, by concentrating the respective information in the low frequency subbands and the "rough" areas by concentrating the respective information in the high frequency subbands. Therefore, the average energy of a subband includes information about the amount of detail in the given direction and resolution (Mallat 1989). The wavelets representation offers an excellent model of how the HVS perceives texture (Tuceryan and Jain 1998). Subband energy formed the basis for several existing feature extraction algorithms in the research literature (Mandal, Idris *et al.* 1999; Smeulders, Worring *et al.* 2000).

There are several approaches with regards to which wavelet filter bank is the most effective for texture analysis. Based on the choice of wavelet transform and whether the image database is semantically homogeneous or not, different filter-sets have been proven to affect differently the accuracy of texture characterization with no clear winner (Brady and Xie 1996; Strang and Nguyen 1996; Lu 1997; Mojsilovic, Popovic *et al.* 2000b; Sam-Deuk and Udpa 2000; Kokare, Biswas *et al.* 2005; Nguyen and Oraintara 2005; Cheng, Law *et al.* 2007). Furthermore, filter banks that offer better PSNRs (Peak Signal-to-Noise Ratios) for higher compression rates do not necessarily offer the best accuracy for texture characterization. In the case of *a priori* compressed images filter selection is unfeasible and, as is discussed in Section 2.7.3, some texture analysis methods attempt to provide accurate characterisation given the constraints of the compression algorithms. For uncompressed images, there are approaches offering a trade-off between texture characterization accuracy and compression efficiency (Wilson and Bayoumi 2003; Su, Hsin *et al.* 2005; Martin, Lukac *et al.* 2006; Pinto Ellas, Villegas *et al.* 2006).

The suitability of wavelets for texture analysis made it the choice for CBIR/CBIC applications in several verticals including: medical imaging and computer aided diagnosis (Muller 2004; Gupta, Lekshmi *et al.* 2006; Dettori and Semler 2007; Gang and Zong-Min 2007; Oliveira, Cirne *et al.* 2007), machine vision for industrial and manufacturing applications (Qiao, Murtagh *et al.* 2004; Han and Shi 2007; Lin 2007; Scharcanski 2007; Zhang and Cheng 2007), forensics and biometrics (Ghouti, Bouridane *et al.* 2006; Nabti, Ghouti *et al.* 2006), adult content filtering (Shih, Lee *et al.* 2007), GIS (Zuyuan and Boesch 2007).

Next, the reader is provided with reviews of the most representative research findings on texture-based image indexing and retrieval in wavelets domain.

2.7 Texture in wavelets domain

As aforementioned, wavelets are an excellent tool for image compression and for texture analysis. To this end, existing research that combines texture analysis and wavelets falls into one of three categories, as follows:

- Texture analysis using wavelets on uncompressed images with no regard to compression.
- CBIR/CBIC systems which perform feature extraction and compression in parallel with some common stages.
- Texture analysis using wavelets on compressed images, typically taking advantage of the characteristics of a specific codec in order to avoid full decompression.

The following sections offer an overview of the most representative research findings with respect to the above taxonomy.

2.7.1 Techniques for uncompressed images

Chang (Chang and Kuo 1993) uses, for feature extraction, a tree-structured decomposition similar to Wavelet Packets (WP) in order to obtain an energy map of a portion a given texture. The energy map is a tree-structured representation of the most significant subbands. Texture is then represented by a dual feature set consisting of: a) the paths (channels) within a tree structure with the most significant

subbands and b) the averaged energies of each node of the path. The significance threshold is chosen arbitrarily based on experimental results. Two methods for training and classification are proposed: one with a fixed number of features and one progressive which eliminates most unlikely candidates first. The classification algorithm performs very well for a set of Brodatz texture images as well as compared to other transform-based methods. The algorithm requires the image data to be in raw format in order to be decomposed using specific wavelet transform parameters. The method is extended in (Chang 1995) for more general pictorial data.

Smith *et al.* (Smith and Shih-Fu 1994) defines sets of texture features using the DWT subbands energy. For a 3-level DWT, feature vectors each containing 10 terms are calculated. Fisher discriminant technique is used to compact the feature vector. Similarity between the vectors of the query and the test images is established with the Mahalanobis distance. The Brodatz texture set is used for the experiments. The authors implement similar techniques in the DCT and pixel domains. The results suggest that wavelets offer superior texture classification performance, in the given framework.

Gabor wavelets have been proved powerful for image processing but not with image compression, mainly due to their complexity. In a representative work on Gabor-based feature extraction, (Manjunath and Ma 1996) decomposed each image into four scales and six orientations. The mean (μ) and the standard deviation (σ) of each subband are used to form a feature vector with 48 coefficients. The feature vectors of two images are compared to determine the level of similarity between them. The six-dimensional decomposition achieves a better directional discrimination at the cost of higher processing requirements.

(Van de Wouwer, Scheunders *et al.* 1999b) proposes the use of directional detail subbands' histogram signatures in order to capture all first order statistics of the wavelet coefficients. The first order statistics obtained via the histogram signatures include: inertia, entropy, local homogeneity and total energy. For further improvement of the accuracy, second order co-occurrence statistics are considered. Although the general performance of the algorithm is well above average, two problems are identified: a) full coverage of all proposed features, which gives the

best results, requires high computational power b) the introduction of the second order statistics does not always improve the performance as some of the tests indicated better classification performance using first order statistics.

In Van de Wouwer (Van de Wouwer, Scheunders *et al.* 1999a) colour and texture information is combined through the use of correlation, specifically covariance, signatures between the wavelet representations of different colour channels. Through a number of tests using different colour spaces and wavelet transforms the authors conclude that colour texture can clearly improve the performance of the gray scale texture-based methods. The methods used are based on Continuous Wavelet Transform (CWT) which cannot be directly applied on compressed images. The best classification performance is achieved in the Karhunen-Loeve space. The latter is formed by the Eigenvector of the correlation matrix of a specific image and is thus image-dependent. However, as the authors note, "as reported by (Ohta and Kanade 1980) the Eigenvector remain approximately the same for a large set of natural colour images". This statement is also confirmed experimentally. Should one desire to process compressed images, K-L space imposes a further limitation. Even thought the JPEG2000 standard permits the definition of the colour space by the codec it is doubtful that the K-L will be massively used due to its poor performance in regards to compression efficiency, and its high complexity.

In (Jacobs, Finkelstein *et al.* 1995) a technique which directly compares the DWT coefficients rescaled versions of images is proposed. In particular, images are rescaled to 128x128 pixels prior to the wavelet decomposition. The metrics that are obtained are: the average colour, the sign and the indices of the M largest magnitude coefficients (a ratio of 40-60 has been used). Finally the feature signatures are arranged into a single data structure for faster access times during retrieval. For existing databases, the technique requires full decompression and rescaling of the images.

Building upon the paradigm of (Bo and Sethi 1996), Wang *et al.* (Wang, Wiederhold *et al.* 1998) have proposed a technique where all images are rescaled to 128x128 pixels and then decomposed into four wavelet layers. Image comparison is performed using a three-stage process. Firstly, the variance of the low-pass *(LL)*

subband of the 4th layer of decomposition is used in order to retrieve 20% of the images. Secondly, the difference between the LL subband coefficients of the 4th level of decomposition between the query and target images is used to further narrow the number of selected images. The final set of images is retrieved by comparison of the coefficients of all the subbands at the highest level of decomposition. A similar procedure is followed for colour images, following a similar procedure on all colour channels. Through this hierarchical matching procedure, this method achieves a low processing overhead. As with the previous example, for existing databases, the technique requires full decompression and rescaling of the images.

In (Hiremath and Shivashankar 2008) texture information is extracted using Gabor filters and the co-occurrence histograms between approximation (LL) and detail (LH, HL, HH) subbands. The joint distributions are obtained using the maxmin composition rule. The subband co-occurrence pairs that are considered by the algorithm are *(LL,LH), (LL,HL), (LL,HH)* and *(LL,abs(LH-HL-HH))* leading to a total of 384 features (hence generated histograms). The images are decomposed using dyadic DWT and the Haar wavelet. Dyadic refers to the fact that each scale differs from the next one by a factor of 2. Tested on 32 Brodatz textures the reported accuracy is superior to other Gabor filters-based techniques.

Based on the fact that, small camera operations (see Figure 1.3) have little impact on the wavelet subband histograms of an image, Mandal *et al.* (Mandal, Aboulnasr *et al.* 1996) have proposed a technique which builds upon (Birney and Fischer 1995). Image similarity is established by comparing the histograms of the *HH, HL, LH* subbands of two images. Because of the high complexity that this technique presents, the comparison of the distribution parameters of the subbands is suggested as an alternative. The probability density functions (pdfs are essentially the histograms) of high-pass wavelet subbands are modelled using the Gaussian density function (GGD) (Birney and Fischer 1995) through the σ (standard deviation) and γ (shape parameter) parameters. The similarity metric between two images (f and g) is, thus, expressed as:

$$d(f,g) = \sum_{k=1}^{Q} B_k \left(\gamma_{f_k} - \gamma_{q_k}\right)^2 + \sum_{k=1}^{Q} A_k \left(\sigma_{f_k} - \sigma_{q_k}\right)^2 \qquad \textbf{[eq 2-1]}$$

where Q is the number of subbands, A_k and B_k are iteratively (through trial and error) established weights for the GGD parameters. The images with the minimum distance are retrieved from the database. In (Mandal, Panchanathan *et al.* 1997), an improved version of the technique is presented, to accommodate variations in image illumination level. To achieve that, scale invariant moments of the histogram are used for estimation of the illumination level. The subband parameters σ and γ of each subband of the target image are then modified to counter the effect of illumination change. Finally, eq. 2-1 can be used for matching images.

In (Do and Vetterli 2002b) a wavelet-based texture retrieval method is presented using Generalized Gaussian density (GGD) to model the distribution of wavelet coefficients. The authors report that the distribution of the coefficients belonging to different subbands can be used to represent textures. The probability density function (*pdf*) of a subband according to GGD is given by:

$$p(x;\alpha,\beta) = \frac{\beta}{2\alpha\Gamma\left(\frac{1}{\beta}\right)} e^{-(|x|/\alpha)^{\beta}}$$

The scheme is efficient in that the GGD model requires only two parameters (*α and β*). This leads to a compact representation compared to histogram-based techniques. For efficiency, the authors propose the use of a maximum-likelihood (ML) iterative estimator to estimate the values of *α and β*. The parameters are estimated for each detail subband of an image that is decomposed using DWT down to three levels. The resulting signature vector has a length of 18. Signatures are compared using a closed form expression of the Kullback–Leibler distance (KLD) between the GGD *pdf*s. For evaluation of the method, 40 512×512 texture images from the VisTex database (MIT 2002) were split into 16 128×128 non-overlapping sub-images, thus creating a database of 640 texture images. The images were also normalised to zero mean and unit variance. The authors report 76.77% average retrieval rate for 3 levels of DWT decomposition. A more recent study by (Teynor, Muller et al. 2006a) reports that the method performs well only for uniform texture regions.

In (Wang and Chang 1996), Wang proposes the use of the correlation function of two spatial domain functions, in terms of their subband coefficients, in the z-

transform domain. Let the low-pass and high-pass analysis filters be $H_0(z)$ and $H_1(z)$, respectively. The correlation of the signal can be expressed as:

$$\widetilde{W}(z)X(z) = \sum_{i,j=0}^{1} F_y(z)\widetilde{Z}_i(z^2)Y_j(z^2) \qquad \textbf{[eq 2-2]}$$

where, $F_y(z) = H_i(z)\widetilde{H}_j(z), i, j = 0,1$

Eq. 2-2 shows that, the correlation of two signals equals the weighted sum of the correlation of their subband coefficients. The authors report that in order to get a reasonable estimate of the correlation peaks, a few subbands with high energy convey enough information, resulting in a reduction in complexity compared to similar pixel-based approaches. The process is still processing intensive with indicative matching times starting at a few seconds for the optimized version of the algorithm.

Wang in (Wang, Jia *et al.* 2001) proposed a region-based image retrieval system called SIMPLIcity. The system segments an image by clustering the feature vectors of similar non-overlapping fixed size 4×4 pixel blocks. Each cluster is treated as a separate region. A greedy algorithm is used in order to aggregate the region matching processes into image matching. Six features are calculated in total, three for colour and three for texture. For texture the system uses the second order moments of the subband-energies. The similarity measure between two images is defined as the weighted sum of distances between region pairs, each pair including one region per image. The weights are calibrated so that regions toward the centre of the image are favoured over regions near the boundaries. Using a set of predefined classes all images in the database are classified using an unsupervised annotation method which is based on low-level feature set. The query is only performed on the images that belong to the same semantic class. The database consists of 200,000 COREL images, and the query is performed using QBE.

Natsev et al presents WaveLet-based Retrieval of User-specified Scenes (WALRUS) in (Natsev, Rajeev *et al.* 2004), a region-based image retrieval system which is also scale, translation and colour-shift invariant. WALRUS considers sliding windows of varying sizes within an image. For each window, a texture signature consists of the LL subband coefficients of the DWT of the specific window. Feature sets are

constructed incorporating colour information. The sets are then clustered in order to group windows with similar characteristics. A cluster identifies a region with homogeneous features and a cluster representative metric (*e.g.* centroid) is used as the characteristic signature of the region. To compensate for the extra processing required for the calculation of the sliding windows features, the authors introduce a dynamic programming algorithm which optimizes reusing of computations. Similarity between images is defined in terms of the fraction of the area of the two images covered by similar regions. Similarity between regions is defined in terms of the distance between their signatures. Compared to SIMPLIcity, WALRUS algorithm favors larger regions over smaller regions instead of central versus boundary. Using a database of 1000 colour images the authors report a retrieval accuracy of over 80% for top-25 retrieval point.

Huang in (Huang, Dai *et al.* 2006) introduce a new wavelet-based feature called Composite Subband Gradient (CSG) vector. Gradient operators are used to detect the magnitude and direction of maximum gray-scale changes from a given point. For *n*-directions, a gradient vector or histogram with *n*-bins (elements), one for each direction, can be generated by summing the respective magnitude values of every point in the image (Haralick and Shapiro 1992). Therefore, the proposed method identifies regions with homogeneous texture by detecting the regional boundaries rather than texture patterns. The authors suggest the use of gradient vectors on DWT subbands after one level of decomposition. A CSG vector is therefore a concatenation of the gradient vectors for the LL, HL, LH and HH subbands. For testing, an image database was constructed using 150 512×512 Brodatz (Brodatz 1966) textures split to 16 non-overlapping images of 128×128 pixels, thus producing 2400 images. Similarity is measured using the Euclidian distance between CSG vectors of two images. The authors report that CSG vectors with 24 bins (*i.e.* 6 directional gradient operators per subband) can achieve retrieval efficacy of 88.51% whereas with 144 bins (*i.e.* 36 directional gradient operators per subband) efficacy rises to 93.2%.

2.7.2 Techniques for compressed images

In (Pi, Tong *et al.* 2006) a method for extraction of feature information from JPEG2000 code-stream is presented. By exploitation of the arithmetic coding

properties of the EBCOT paradigm (Lian, Chen *et al.* 2003) adopted by JPEG2000, the proposed method creates a Bit Plane (BP) signature for each image code-block. Arithmetic coding in EBCOT involves the generation of scalable multi-pass bit-streams per code-block designed for optimal rate-distortion and signal-to-noise ratio (SNR) representation of the image. The BP signature is defined as the product of Bernoulli distributions of the high-pass subbands bit planes generated by the arithmetic coder. The product of Bernoulli distributions serves as an approximate model for the energy distribution of subband coefficients. Aggregation of BP signatures for three passes (also called *quality layers*) of the EBCOT arithmetic coder results in the *three-pass layer probability* (TPLP) texture feature descriptor. The descriptor is tested using a database of 32 Brodatz textures and the Daubechies (9,7) and Daubechies (5,3) filter banks. The reported retrieval accuracy results outperform other wavelet-based approaches such as (Do and Vetterli 2002b; Mandal and Liu 2004).

Descampe *et al.* (Descampe, Vandergheynst *et al.* 2006) present a texture-based retrieval technique using a set of cascaded classifiers. The classifiers extract texture information from either the headers or the wavelet coefficients of JPEG2000 compressed images. The technique starts by extracting a number of low-complexity features from image headers and gradually cascades more complex wavelet-based features. The header extracted features are the source entropy and maximum modulus values which are readily available in JPEG2000 compressed images. The wavelet-based extracted features include a set of bit-plane derived histograms for all available DWT resolutions as well as a set of descriptors based on the Generalized Gaussian Density (GGD) modelling of the wavelet coefficients distribution. The aim is to gradually reduce the image sample pool using low-complexity techniques at the early stages so that only a small subset of the pool needs to be processed through the more complex bitmap histograms and GGD-based classifiers. Three bitmap histogram and GGD-based classifiers are built each analysing more a larger number of bit planes, *i.e.* including finer detail information about the image. The technique was tested on a database of 640 128×128 gray scale texture images belonging to 40 texture classes. In total, 7 different classifiers are cascaded each one adding in complexity. Each successive classifier rejects and discards a number of images from the sample pool until the last classifier which keeps 15 images. The sensitivity of

each consecutive classifier, *i.e.* the number of images discarded at each step, has been determined empirically. The authors report that the proposed cascaded technique performs comparably to the more complex GGD-based classifiers for a lower processing cost. The savings in processing cost are reported in terms of the amount of data that needs to be processed. Specifically the proposed method processes approximately 56% of the header data and 17% of the decoded bit planes compared to the full GGD-based classifier. Nevertheless, the GGD modelling parameters calculation is generally regarded as a computationally expensive process.

In (Au, Law *et al.* 2007) the authors exploit the similarities of the spectral responses between BDCT (the basis of JPEG standard) and DWT (the basis of JPEG2000 standard) in order to identify a common set of statistical metrics. The aim is to use those metrics in order to construct a feature set which is applicable to both JPEG and JPEG2000 compressed images. The authors propose a way of rearranging and concatenating the DC and AC coefficients of the BDCT blocks in groups so that each group maps directly onto the spatial-frequency characteristics of each DWT subband. The statistics that are used for feature information extraction are the energy signature and the 2^{nd} and 3^{rd} order normalized central moments of the Significance Maps. Both statistics have translation and rotation invariant characteristics although these characteristics are not evaluated or benchmarked in the paper. The resulting feature signature is a 35-dimensional vector. Similarity is tested using the L_2-distance. The method is tested on an image database of 1800 images belonging to 9 pre-defined classes of variable detail and texture distribution. To assess the retrieval performance, a result is considered to be correct if both the query and the retrieved images belong to the same class. The authors report that the proposed method has above-average retrieval performance for image classes that have more lower-frequency information (smooth areas and large distinct objects) and below-average retrieval performance for those that have more high-frequency information (textures). It must be noted that although the system is evaluated as a generic retrieval system, the normalized central moments are commonly used for shape information characterisation rather than for texture (Liang and Kuo 1999; Saber and Tekalp 2006). The authors also report that the uniformity between the BDCT and the DWT-based feature performances is also more apparent for lower frequency transform coefficient blocks.

Mandal and Liu (Mandal and Liu 2004) propose a retrieval technique that extracts feature information from JPEG2000 code block bit-planes. A map of the location of significant bits on each JPEG2000 bit-plane is generated. The map is used as a descriptor of the pixel energy distribution across the image. Another vector descriptor is introduced in order to improve the performance of the algorithm. The descriptor includes variance and mean of the numbers of bit-planes per individual code-block. The features are extracted separately for each of the RGB colour channels and then fused, therefore creating a colour-texture signature. The method is capable of offering progressive detail feature extraction from JPEG2000 compressed images although this requires that the images are compressed (or re-compressed) using specific parameters. The reported performance tests are based on an image database of 3000 images consisting of groups of 3 semantically similar images, whereas the smallest employed retrieval window is 60 images. Further experiments with larger groups of similar images are required for more conclusive results.

In (Teynor, Muller et al. 2006b) a technique which extracts colour and texture information from JPEG2000 compressed images is presented. Colour information is extracted using a 5×8×8 (320 bins) histogram from the two chrominance (C_b C_r) components of a wavelet decomposed image. The texture descriptor is constructed using the Gaussian Mixture Model (GMM) to model the wavelet coefficient distribution of each subband. The similarity or distance between two descriptors is calculated using the Kullback-Leibler divergence (KLD) or relative entropy (Do and Vetterli 2002b). Due to the high complexity of KLD the authors use an approximation technique described in (Goldberger, Greenspan *et al.* 2003). The proposed texture feature vector comprises 72 coefficients for images decomposed down to 5 levels. The proposed methods were tested for classification performance against a database of 4501 images. In order to perform texture classification experiments ground truth was established for a set of 11 randomly selected images with the aid of three independent users. The authors report that the method is more accurate on non-uniform texture images than methods which employ Generalized Gaussian Densities (GGD) (Do and Vetterli 2002b) for an extra cost in performance. Reported retrieval times indicate an increase of 50% in time (900ms over 600ms) for GMM over GGD. It

must be noted that this increase does not include the time for the colour feature extraction.

2.7.3 Combined compression & texture characterization

An alternative approach to partial decompression is the computation of feature parameters in parallel to the compression process. Such approaches are not applicable to databases containing already compressed images. The combined feature extraction and image compression techniques trade off compatibility with compression standards to retrieval speed and performance.

In Waveguide (Liang and Kuo 1999) Liang *et al.* presented a combined image compression and retrieval system based on the three primary low-level features colour, texture and shape. All the content feature information is extracted in parallel to the compression process and is stored separately in a signature database. Using subband energy and Significance Map information, Liang *et al.* fuses information about texture, shape and colour in order to create a single signature to characterize the content of an image. Texture content information is extracted using the normalized significance of the subbands of a dyadic wavelet transform. The significance is first determined at coefficient level using an experimentally established threshold. For a predetermined threshold value T the significance N_i of the i^{th} subband is measured according to the number of significant coefficients, as follows:

$$N_i(T) = \left|\{c_j \mid c_j \in S_i, c_j \geq T\}\right| \qquad \textbf{[eq 2-3]}$$

where S_i denotes the i^{th} subband and c_j the magnitude of the j^{th} coefficient. The importance of the subband is normalised according to:

$$N_{norm,i}(T) = \frac{N_i(T)}{\sum_i N_i(T)}, i = 1,2,\ldots,n \qquad \textbf{[eq 2-4]}$$

The texture vector TV is then composed by the normalised values of subband importance of the subbands:

$$TV = \{N_{norm,1}(T), N_{norm,2}(T) \ldots N_{norm,n}(T)\} \qquad \textbf{[eq 2-5]}$$

In order to compare two images with respect to their textures, Waveguide uses the L_1-distance of their vectors. A weighting factor is also used to compensate for the higher energy concentration in higher levels of decomposition.

Fahmy *et al.* (Fahmy, Black *et al.* 2006) exploit the spatial frequency content of images in order to produce a method for joint texture compression and classification. Based on the human visual system, the method measures the degree to which an image contains coherent (perceptible) phase information. This is achieved via the spatial frequency phase (SFP) texture parameter. SFP represents the vertical and horizontal position of edges within a local region. In this context, SFP is realized by applying Laplacian-of-Gaussian (LOG) filters to the LH and HL subbands of a DWT decomposed image in order to detect points of phase coherence, also known as coincidents. Texture is characterized by correlation of the number of the coincidents through 16 different spatial frequency channels. Calculation of the coincidents is a computationally intensive process. The reported performance of the algorithm is promising albeit it is restricted to user evaluation tests.

A modified version of the SPIHT compression algorithm (Said and Pearlman 1996b) aimed to unsupervised segmentation/classification applications is presented in (Chang and Carin 2006). The technique is focusing on remote sensing applications where large images have to be transmitted remotely at low bit rates and segmented. After wavelet decomposition and prior to compression, the proposed method performs texture-based segmentation. The importance of the wavelet coefficients is then scaled in the context of realizing this segmentation. The aim is to upscale those coefficients that increase the accuracy of the segmentation/classification task. In the context of SPIHT, upscaled coefficients will appear earlier in the bit stream. The mean-square error (MSE) is also accounted for in the final importance weighting, similarly to SPIHT, to accommodate for the imperfections of the classifier. Within the context of image segmentation, the authors report that the performance of the proposed classifier (in terms of the average error rate) is superior to two benchmarks namely, relevance vector machine (Tipping 2001) and linear least squares regression model (LSRM) (Hastie, Tibshirani *et al.* 2001).

2.7.4 Rotation Invariant Texture characterisation

The abovementioned feature characterization techniques assume very limited camera movement for the test data (see Figure 1.3). The accuracy drops abruptly when a rotated version of a texture is processed by the system. To address the issue, a number of techniques appeared in research literature, which can efficiently

characterize texture, even if the sample and prototype images are rotated versions at different angles

In Qi et al (Qi and Shen 1994) rotation invariance is achieved by defining the mother wavelet in polar coordinates thus resulting in a complex wavelet transform. The algorithm is applied on a set of English character images where it outperforms existing rotation invariant counterparts, such as complex Zernike moments-based methods (Chong and Raveendran 2003).

Rashkovskiy *et al.* (Rashkovskiy, Sadovnik *et al.* 1994) have proposed a feature extraction method which is unaffected by variations in shift, rotation and scale (SRS). A class of non-linear wavelet transforms is used, in which the mother wavelet is adapted for every input to achieve the invariant characteristics. The wavelet parameters are calculated iteratively and per application. The method is processing intensive and by using non-linear wavelets it is unsuitable for use on already compressed images.

Porter and Canagarajah (Porter and Canagarajah 1997) suggest a number of techniques for rotation invariant texture classification. One wavelet-based, one Gabor filter-based and one GMRF-based scheme. According to their findings the wavelet-based technique performs the best, in both rotation and non-rotation invariant setups. Porter develops the invariant texture analysis by first decomposing the image into ten channels (three-level DWT decomposition). The L_1-norm (the sum of absolute values of each coefficient) of each subband is used in order to construct the feature vector. The diagonal detail subbands are excluded since, according to the authors, they cause deterioration of the ability of the texture descriptor. Rotation invariance is achieved by *combining pairs of diagonally opposite wavelet channels to form single features.* The method is tested on a database of sixteen Brodatz textures at six orientations. The authors report that the main drawback of this method is that the directional information is lost when the channels are combined. This accounts for most of the misclassifications which are reported to be 4% on average. The scheme uses the Mahalanobis classifier for classification.

In (Van de Wouwer, Scheunders *et al.* 1998) the issue of rotation invariance is addressed by using non-separable wavelet transforms, instead of separable DWT, which is commonly used in wavelet compression codecs. Using isotropic wavelets, Wouwer et al presents methods for rotation invariant texture classification and image segmentation. The results for classification are comparable to those of (Porter and Canagarajah 1997; Ojala and Pietikainen 2002).

Jafari-Khouzani in (Jafari-Khouzani and Soltanian-Zadeh 2005b) tackles the issue of rotation invariance in texture classification with a technique that combines Radon transform for directionality detection and wavelets for texture feature extraction. Radon transform is used to detect the principal direction of anisotropic textures. The frequency components of anisotropic textures (also called directional) hence the wavelet features change considerably as the image rotates. After applying Radon transform the image is rotated along the orientation of the detected principal direction. The image is then decomposed using the Daubechies db_4 filter. The extracted texture features are in terms of statistics of energy per resolution per subband and the non-uniformity of subband coefficient values. The authors report that the algorithm is robust to additive white noise and illumination variations. Experiments are performed using selected Brodatz textures. The reported results indicate that the technique has comparative performance to the benchmark $LBP_{P,R}$ algorithm (Ojala, Pietikainen et al. 2002a) albeit at a higher processing cost imposed by the directionality detection and rotation of the unclassified images. The scheme is also not compatible with any compression standard.

In (Kokare, Biswas *et al.* 2007) the authors describe a rotation invariant wavelet-based texture retrieval method. The proposed method introduces a set of 2-D rotated wavelet filters (RWF) based on the Daubechies eight-tap filters. The authors report that Daubechies filters are more capable that Haar filters in retaining high frequency edge information due to their overlapping window structure. The 2-D RWF are used to calculate a non-separable 2-D DWT which is aligned to 45° compared to the original image. The texture signature is constructed by calculating the energy and standard deviation of each subband. For retrieval, the Normalised Euclidian Distance and the Canberra distance metrics are tested. The authors do not report what is the comparable effect of the two distance metrics in the accuracy or the

speed of the retrieval. Notably, the Normalised Euclidian Distance requires pre-processing of the whole image database in order to determine the mean and the standard deviation values for each subband. The authors report that combination of (DWT + RWF) and (DWT + a subset of RWF derived features) leads to relatively small signature sizes (60 to 80 coefficients) and yields average retrieval accuracy of approximately 80% for the two databases. The methods are tested using 108 Brodatz textures (Brodatz 1966) split into 1856 sub-images and 40 VisTex textures VisTex (MIT 2002) split into 640 sub-images.

The issue of rotation invariance is addressed in (Sastry, Pujari *et al.* 2004) using as a starting point the wavelet decomposition and a set of radial symmetric filters. The filters are used in order to obtain a non-standard wavelet decomposition of the image in order to extract rotation invariant texture features. The features are generated by calculating the means and variances of L concentric circular regions within the LL, HL, LH and HH subbands. The method is tested for 2 levels of decomposition and 4 concentric regions, *i.e.* the size of the feature vector is $2\times4\times7 = 32$ coefficients. The authors report that the generated feature vectors were experimentally confirmed to be robust to rotation. To speed up the retrieval process the authors suggest clustering of the complete image database using the k-means clustering algorithm (Pujari 2001) which is based on feature vectors. This way the retrieval process is performed in two stages, the first one to eliminate irrelevant clusters and the second one to perform full query in the relevant clusters. However, the clustering process requires pre-processing of the entire database.

In (Muneeswaran, Ganesan *et al.* 2005) a rotation and scale invariant texture classification scheme is presented called combined invariant feature (CIF). CIF is compiled using information extracted by crude wavelets, like the Gaussian and Mexican Hat filters and orthogonal wavelets like the Daubechies filters. For the orthogonal wavelets sub-feature, the image is transformed using wavelet packets decomposition. Several subband energy measures are concatenated such as the norm-1 and norm-2 energies, the standard deviation, the average residual and the entropy. Similarly, first and second order statistical properties of the Gaussian and Mexican Hat wavelets are calculated as texture descriptors. Rotation invariance is achieved by extracting the non-invariant features at several angles and creating a

composite feature signature. Variance analysis is employed in order to reduce the extracted feature space to only those that have higher discrimination ability. The method was tested on a database of 105 256×256 texture images which were split into 16 non-overlapping 64×64 blocks. Half of the blocks were used for training and half of them ware used for testing. Rotated versions of the blocks at various angles were generated resulting in a database of 1680 images. The Euclidian distance between the test image feature vector and the different classes feature vectors was used for classification. The authors report 90% average classification accuracy for the CIF combined feature.

A rotation invariant texture classification technique based on the ridgelet transform is presented in (Pan, Bui *et al.* 2008). A three-step process is used in order to extract texture information. The Radon transform is used to obtain a *k*-orientations polar frequency (Fourier) representation of the image. B-spline wavelets are applied to perform 1-D wavelet transform on each of the *k*-orientations thereby obtaining a 2-D frequency-orientation decomposition of the image with six subbands at each orientation. Several orientations are grouped together to reduce the polar dimensionality. For each subband the first and the second order moments of the coefficient amplitudes are calculated as texture features. Applying a 1-D DFT on the vectors of 1^{st} and 2^{nd} order moments generates the shift invariant features. Shift invariance in polar coordinates corresponds to rotation invariance. The k-nearest neighbour classifier was used for feature classification. The method was tested using a number of rotated image data sets generated from the Brodatz (Brodatz 1966) album and VisTex database (MIT 2002), and compared with similar approaches (Fountain and Tan 1997; Jafari-Khouzani and Soltanian-Zadeh 2005a).The authors report correct classification rate of over 97%.

2.8 Similarity metrics

In the context of image retrieval and classification, a query is a two-stage process: a) measuring the similarity between images (or images and image classes for classification systems) and b) processing of the similarity results (*e.g.* ranking or classification/segmentation). Processing wise, similarity measurement is the most demanding stage of the two. The speed of similarity calculation is affected by the size of the database, the hardware, the size and format of the feature signatures and

the distance calculation method. The overall efficiency of a similarity calculation technique is also characterised by the effect that it has on the accuracy of the results. When feature signatures are in the form of vectors or histograms, image similarity is quantified as the distance of the signatures. The distance metrics that are most commonly used for feature signature comparison include: Euclidian (L2) distance, Manhattan (L1) distance, histogram intersection, Hamming distance, Quadratic distance and Mahalanobis distance. The majority of these are based on Euclidean space and have similar retrieval efficiency (Smith 1997).

In (Zhong and Defee 2007) the authors evaluate several distance metrics used for comparison of histogram-based feature signatures. These include: the L_1-norm, the L_2-norm, the Standard Euclidian (SE) distance, the Cosine (Cos) distance and the Correlation (Cor) distance. The authors test the system using the Colour FERET database (circa 2003) containing 10000 images of human faces maintained by the US National Institute of Standard and Technology (NIST). For feature extraction the authors use the AC coefficients of an image encoded using the ITU-T H.264 standard. Specifically, a 3×3 ternary (tri-state) matrix is created by thresholding a central coefficient and the surrounding AC coefficients resulting in a 6560-bins histogram. The histograms of three AC coefficients (3x6560 bins) comprise a single feature histogram. The authors report that for the specific feature the L_1-norm has superior performance and lower processing cost than the other metrics. The L_1-norm (or Manhattan norm) has been adopted extensively as a signature distance metric by researchers (Liang and Kuo 1999; Jiang, Liu *et al.* 2003; Wilson and Bayoumi 2003; Hadjidemetriou, Grossberg *et al.* 2004; Schaefer 2004; Cheng, Law *et al.* 2007; Cheng, Chien *et al.* 2008)

An example of measuring the similarity of two images based on the principles of the Correlation distance is presented in (Wang, Bovik et al. 2004). Starting from pixel data, similarity is established by comparing luminance, contrast and structure. Luminance is expressed by the mean intensity of the image and contrast is expressed by the standard deviation. The image structure is then expressed by the correlation or inner product of luminance and contrast. Three comparison functions are created, one for each of the features. These are then combined to yield the overall similarity measure called the Structural SIMilarity (SSIM) index. The SSIM

index is primarily designed for image quality assessment. This involves the comparison of an original undistorted image versus a distorted (e.g. compressed at low-bit rates, blurred, with added artificial noise, contrast stretched) version of the same image. The SSIM index is, thus, a quantitative measure of the level of perceived quality drop. The experimental results indicate that the SSIM index offers a more accurate and consistent image quality assessment when compared to several other quality assessment models (PSNR, Sarnoff, UQI). The authors report that the index can be adapted for other image processing applications such as rate-distortion optimization for the design of image compression algorithms albeit at a higher processing cost compared to the widely used Mean Square Error (MSE). Furthermore, by incorporating more feature descriptors the index can be extended to content-based image retrieval applications.

The nearest neighbour (NN) is the classification rule based on the Euclidean distance (Fukunaga 1990). The NN search compares the query vector with each class template vector individually, that is, it sees a class as consisting of unrelated points in the feature space. The k-NN classification rule is an extension of the NN rule which *"... involves finding a hypersphere around a point x which contains K points (independent of their class), and then assigning x to the class having the largest number of representatives inside the hypersphere ..."* (Bishop 1995). The k-NN classification rule is commonly used for image classification applications (Van de Wouwer, Scheunders *et al.* 1999b; Jafari-Khouzani and Soltanian-Zadeh 2005a; Hiremath and Shivashankar 2008; Pan, Bui *et al.* 2008).

Despite the several indexing schemes intended to provide efficient data structures for storing signatures, attempts to speed-up the calculation of the popular Euclidian and Manhattan distances are scarce in the literature. Here follow two of the most representative examples.

In (Berman and Shapiro 1999) the authors present a system which uses an indexing scheme and algorithm based on the triangle inequality. The system can often return matches by comparing the query image to only a percentage of the database. Given a database of N images represented by $I_x \;\; x \in [1, N]$ and a query image Q, M key images represented by $K_p \;\; p \in [1, M]$, can be selected from the database. Considering

all the three distances among the images from the database, the query, and the key images, a triangular inequality is formed as follows.

$$d(I,Q) \geq \max \left| d(I,K_p) - d(Q,K_p) \right| \qquad \textbf{[eq 2-6]}$$

This provides a lower-bound for the distance between all the images inside the database and the query image. Since all $d(Q,K_p)$ values can be pre-calculated, p number of distance calculations are required in order to establish the right hand side of eq.2-6. With careful selection of key images the number of distance calculations can possibly be reduced. The savings from the technique are non-deterministic as they highly depend on the distances between the query and the key images.

Wilson (Wilson and Bayoumi 2003) describes a method which enables the calculation of L_1- and L_2-distances by successive refinement during the Embedded Zerotree Wavelet (EZW) decoding process. To achieve that, the authors introduce an extra end-of-subband symbol to the code stream. The technique uses the significance information contained in the bit-stream in order to reduce the number of calculations required for the calculation of the L_1- and L_2- distances. By introducing the extra symbol, the method is also capable of achieving comparable classification accuracy at a lower bit-rate than full-decompression methods. Classification performance is tested using the sum-of-squares and the L_1-norm subband energy features. The authors report significant improvements (over 75%) in memory and number of calculations requirements over full-decompression and processing. However, the magnitude of improvement depends on the nature of the image (*i.e.* the content and structure of the bit-stream) and the selection of distance metric (L1 or L2).

2.9 Other contributions

This section includes two contributions on texture analysis which do not fall into any of the above categories but are related to this work.

Ojala et al (Ojala and Pietikainen 2002) proposed a pixel-based technique for texture classification with excellent characteristics. Ojala's simple concept yet elegant and effective solution is still considered a state-of-the-art texture analysis technique (He, Li *et al.* 2007; Melendez, Puig *et al.* 2007) and has been adopted for applications mostly in medical imaging (Caicedo, González *et al.* 2007; Unay, Ekin *et al.* 2007)

and face recognition (Huang, Wang *et al.* 2007; Park and Kim 2007; Tan and Triggs 2007). The algorithm has been used by several researchers as a benchmark (Jafari-Khouzani and Soltanian-Zadeh 2005b; Miguel Angel, Dom *et al.* 2007). The feature extraction algorithm is based on the Local Binary Pattern (LBP) operator, an autoregressive occurrence model formed by a circular multi-scale and multi-point pattern around a reference pixel. The calculation of the feature is based on the relevant intensity of each point in the pattern compared to the central reference point. Based on LBP, a generalized grey scale and rotation invariant operator, which detects uniform patterns in circular neighbourhoods of any quantization and at any spatial resolution, is developed. The operator exploits the fundamental properties of texture such as edges and spots. The variance of the LBP operator is used as an additional operator which is immune to variations in contrast. Both operators are combined in different spatial resolutions to achieve scale invariance. For the classification process, a non-parametric principle based on the log likelihood ratio is used.

The quad-tree structure is central to the design of wavelet-based codecs (Shapiro 1993). Interestingly, Quad-trees have been used as a means of non-wavelets-based texture characterization in (Vassilakopoulos, Manolopoulos *et al.* 1993; Lin 1997; Poulakidas, Srinivasan *et al.* 1997; Annabelle, Patrick *et al.* 1999). In (Smith and Chang 1994) Quad-trees are used in order to break down an image to several layered regions at pixel level. Each region is then transformed using DWT and the mean absolute value and variance measures are calculated as texture descriptors. The wavelet derived features are used in order to determine texture similarities between regions at the same level of the quad-tree break down and between parent-leaf nodes, hence performing crude image segmentation. Therefore, in this example Quad-trees are not used in the context of wavelets but instead as a per-processing step in pixel domain. Quad-trees have also been used for feature extraction in Multi-resolution Fourier Transform (MFT) domain (Wilson, Calway *et al.* 1995).

2.10 Summary & critical review of existing research

Research on the mathematical characterization of texture begun in the early seventies. Through the years, research interest on the subject has grown as new techniques and applications with more complex requirements appeared. Recent studies indicate that texture characterization accuracy needs to be improved for wider adoption by practical applications. (Section 2.3)

Recent efforts have focused on wavelets for several reasons such as the multi-scale spatial-frequency representation which corresponds to the HVS. Also wavelets have been adopted as the de-correlation stage of state-of-the-art compression algorithms. The spatial-frequency representation of wavelets is very convenient for texture analysis as the energy of the subbands conveys information about the "smooth-ness" or "rough-ness" of specific image regions, resolutions and orientations. (Section 2.6.4)

The remaining of this Section offers a critical review of the existing research which was presented earlier in this chapter. This critical review is categorized in four "themes" which highlight the identified shortcomings that will lead to the overall conclusions and the problem statement in the last section of this chapter. These themes are:

- Accuracy of wavelet-based texture descriptors
- Rotation-invariant wavelet-based texture descriptors
- Feature extraction from compressed images
- Speed of similarity distance calculation

ACCURACY OF WAVELET-BASED TEXTURE DESCRIPTORS

Given the nature of the wavelet transform, discussed in Section 2.6.4, it is not a surprise that many wavelet techniques for uncompressed images are constructing texture descriptors around the first and second order statistics of subband energies (Chang and Kuo 1993; Smith and Shih-Fu 1994; Manjunath and Ma 1996; Wang and Chang 1996; Van de Wouwer, Scheunders *et al.* 1999b; Hiremath and Shivashankar 2008). Other approaches incorporate colour (Jacobs, Finkelstein *et al.* 1995; Van de Wouwer, Scheunders *et al.* 1999a), do one-to-one coefficient comparison of LL

subbands at low-resolutions (Wang, Wiederhold *et al.* 1998), use model-based techniques such as GGD for modelling of the coefficient distributions (Mandal, Panchanathan *et al.* 1998; Do and Vetterli 2002a) or move from global image texture to regional texture (Wang, Jia *et al.* 2001; Natsev, Rajeev *et al.* 2004; Huang, Dai *et al.* 2006).

As discussed in Sections 1.2.2.2 and 2.5.1, a fundamental characteristic of texture is the repetition of primitive patterns (texels). Exploitation of this characteristic have been proven very effective for characterizing texture in pixel domain, with prominent example the LBP algorithm (Ojala, Pietikainen et al. 2002b) which employs the autoregressive occurrence model and is still considered as one of the state-of-the-art algorithms. As aforementioned, in wavelets domain most of the techniques have focused on the 1st and 2nd order statistics with very few exceptions which take into consideration local intra-subband information down to coefficient level.

Examples from the literature survey are: (Van de Wouwer, Scheunders *et al.* 1999b) which suggest 1st and 2nd order statistical analyses of coefficient co-occurrence within a subband and, (Hiremath and Shivashankar 2008) which compares co-occurrence of coefficients between two images and then uses 1st order statistics (*e.g.* mean of the absolute deviations). In the most relevant example (Huang, Dai *et al.* 2006) uses cumulative gradient vectors to create a histogram of energy distributions for all the subbands. In either case, studies report that approaches based on the LBP paradigm outperform gradient vectors (Ojala, Pietikainen *et al.* 1996) and co-occurrence matrices (Melendez, Puig *et al.* 2007; Zhao and Pietikainen 2007) in terms of accuracy and processing requirements. The good performance of the co-occurrence and gradient-based approaches offer an indication that intra-subband analysis can improve the accuracy of texture analysis. To further validate that no work has been done in the specific area, a search was performed for the combination of keywords *texture*, *wavelets*, *autoregressive* and *occurrence* in the ACM Digital Library, IEEE Xplore, Elsevier ScienceDirect and SpringerLink portals (Apr 2008) and yielded no relevant or comparable works.

Another important finding is that although the quad-tree structure is central to the design of wavelet-based codecs (Daubechies 1992; Shapiro 1993), including the

commercially available SPIHT (Said and Pearlman 1996a) and JPEG2000 (Rabbani and Joshi 2002b), there have been no attempts to directly exploit the quad-tree nature of subband coefficients as a texture descriptor in wavelets domain. Notably quad-trees have been successfully employed in pixel-domain and in MFT-domain (Multi-resolution Fourier domain; see Section 2.9).

ROTATION-INVARIANT WAVELET-BASED TEXTURE DESCRIPTORS

Rotation invariance in texture characterization is a major *sensory gap* issue and researchers have proposed methods with minimal effect to their retrieval accuracy when the orientations of the images that are being compared do not match (Section 2.7.4). Nevertheless, as discussed in Chapter 1, rotation invariance of texture descriptors remains an open issue. Most wavelet-based techniques that are addressing the issue of rotation invariance are employing rotated filters resulting in non-separable wavelet transforms (Section 2.7.4). The problems with non-separable wavelet transforms are that a) the data produced are more than the separable DWT, also resulting in higher processing cost, b) none of the approaches are compatible with existing compression algorithms. From the conducted survey the only exception is (Porter 1997) which investigates the use of DWT for rotation invariant texture which according to (Ojala, Pietikainen et al. 2002b) has inferior performance compared to an LBP-based rotation invariant texture descriptor.

FEATURE EXTRACTION FROM COMPRESSED IMAGES

Existing literature refers to "working compressed domain" for systems which are capable to process transformed coefficients (point C in Figure 1.2). Although this statement is arguable since there is still considerable processing required to obtain the transform coefficients when the initial data is the compressed bit-stream (*i.e.* from point E to point C in Figure 1.2). Chapter 5 elaborates on the limitations of considering transform coefficients as "compressed domain" and introduces a new technique which extracts features directly from the compressed bit-stream. In this section the convention of point C in Figure 1.2 being the threshold between "compressed domain" and "non-compressed domain" is followed in order to qualify the respective feature extraction approaches.

Considering both image retrieval and compression in wavelets domain it is possible to categorize the texture feature extraction approaches to: a) those which do not consider the compression problem (Section 2.7.1), b) those which suggest a joint compression and indexing/retrieval solution (Section 2.7.3), and, c) those that attempt to combine the two problems by extracting features directly from compressed images (Section 2.7.2).

The inability of those techniques which do not consider the compression problem to be applied to compressed domain is because at least one of the following conditions is true: pre-processing at pixel domain is required, special transformations (*e.g.* Gabor, CWT) are employed which are incompatible with existing compression algorithms, specific DWT parameters (*e.g.* filter banks, levels of decomposition) are required which are incompatible with existing compression algorithms.

In the cases where pre-processing in pixel domain is required (e.g., (Sastry, Pujari et al. 2004; Kokare, Biswas et al. 2007)) a database of images which is already compressed will have to be entirely decompressed in order to perform the pre-processing stages. Referring to the diagram of the generic transform codec shown in Figure 1.2 all the stages from E to A (or right to left according to the diagram since this is decompression) must be executed. The process will need to be repeated for each new image that is introduced in the database. Note that an additional shortcoming of this approach is that pre-processing of the entire database must be repeated every time there is an alteration of the attributes of the extracted features or a new feature is introduced.

In case where transformations such as Gabor or CWT are employed (*e.g.* (Van de Wouwer, Scheunders et al. 1999a; Hiremath and Shivashankar 2008)) which are incompatible with existing compression algorithms the system will have to decompress the image at least to the point where the inverse transformation is performed (*i.e.* from point E to point B in Figure 1.2). Depending on whether the system has provision for processing image data in the same colour space as the one that was used for compression an additional stage of processing might be required (*i.e.* from point B to point A in Figure 1.2) in order to get the data in a suitable form. Therefore systems that use incompatible transformations with compression codecs

do not qualify as working in "compressed domain" following the rule that point C in Figure 1.2 is the threshold.

When specific DWT parameters are required (*e.g.* (Wang, Wiederhold et al. 1998; Mandal and Liu 2004; Natsev, Rajeev et al. 2004)), for example, specific filter banks or levels of decomposition, and these parameters differ from those used during compression, then the transform coefficients that are made available in point C of Figure 1.2 require further processing before performing feature extraction. In the case of different filter banks the images must be decompressed up to the point of inverse transformation (point B in Figure 1.2) and re-transformed using the required filters. If the difference is a mismatch between the required level of decomposition for feature extraction and the one that was used during compression then, disregarding other possible mismatches like filter-banks and subband sizes, there are two cases: a) the compressed image is decomposed to more levels than those required by the feature extraction system, b) the compressed image is decomposed to fewer levels than those required by the feature extraction system. In case (a) if the feature extraction system relies on the coefficients of the LL subband then inverse DWT (iDWT) must be performed up to the desired level. In case (b) the DWT tree must be further decomposed to the required level. Mapping the two cases to the diagram of Figure 1.2 results in a location between points B and C. The issues are similar if specific subband sizes or subband aspect ratios are required. Therefore, none of these cases qualify as "compressed domain" according to the threshold of point C.

The methods which suggest joint compression and feature extraction (such as those described in Section 2.7.3) fall into one of two categories: a) those that pre-process the entire database and store feature information separately, b) those that either use special compression schemes that make feature extraction possible without requiring full decompression. Systems that fall into category (a) do not qualify as "compressed domain" since the source data for feature extraction is essentially in pixel form. Systems of category (b) do qualify as capable of working in "compressed domain" but suffer from the following shortcomings:

- The entire image database needs to be re-compressed;

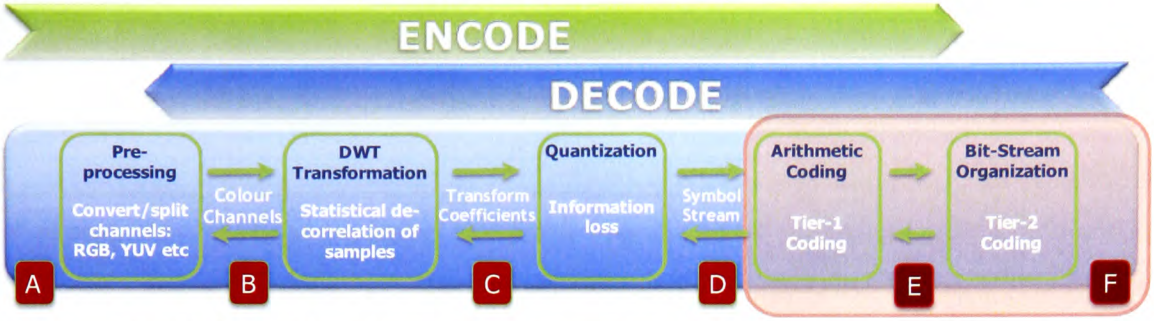


Figure 2.2: JPEG2000 fundamental building blocks. Adapted version of fig. 1 in (Rabbani and Joshi 2002b).

- Any alteration or amendment to the extracted features requires re-processing of the entire database;
- The processed images cannot be manipulated by standard image processing systems as they can only be decompressed using the specific specialized codec.

Research in feature extraction directly from compressed images is monopolized by JPEG2000. All techniques extract information from the bit planes while two of the techniques also use information about the significant coefficients distribution that the JPEG2000 file format requires to be stored at the image headers (Mandal and Liu 2004; Descampe, Vandergheynst *et al.* 2006). However, most of the proposed methods require the full inverse DWT to be reconstructed in order to extract the necessary feature information. One exception is (Mandal and Liu 2004) which also offers progressive detail feature extraction provided that specific compression parameters have been used during compression. Another exception is (Pi, Tong *et al.* 2006) which does not require full reconstruction but requires at least three quality layers to be extracted for each code block.

It must be noted that the JPEG2000 file format employs a number of variable-length "containers" called *boxes* in order to facilitate the encapsulation of metadata and offer flexibility (*e.g.* multifunction files and support for streaming) and expandability (Houchin and Singer 2002). Furthermore, the JPEG2000 coding and bit-stream organization schemes partition data in order tiles, precincts and code-blocks aiming to more flexibility in applications requiring random access and individual compression parameters. This introduces an extra stage/block (see Figure 2.2), called "Tier-2

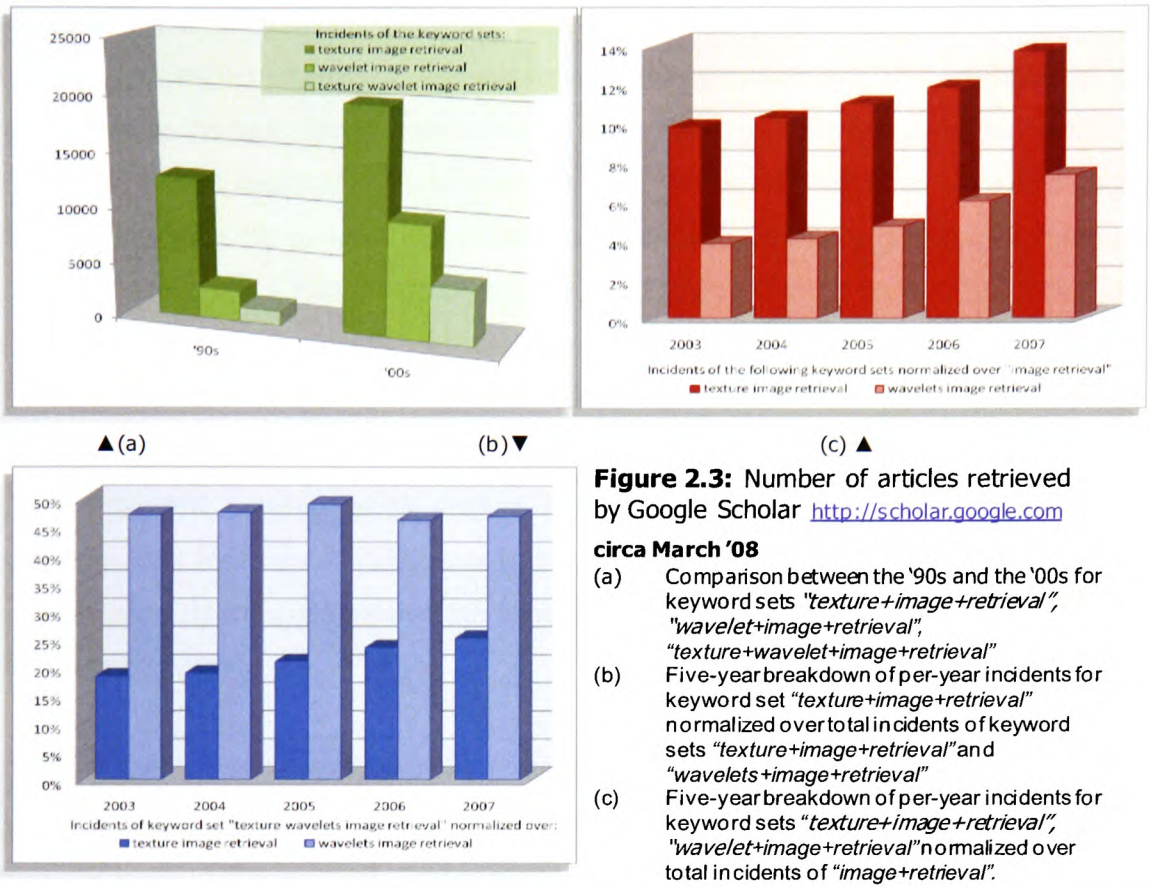
coding" (Rabbani and Joshi 2002b), with respect to the typical codec diagram of Figure 1.2.

Summarizing, feature extraction in JPEG 2000 compressed domain poses a number of limitations, one of them being the extra Tier-2 Coding stage which is an inevitable step for reconstruction of the bit-planes which contain wavelet coefficient information. Although some feature information can be extracted from the image headers, the systems have to include subband derived features in order to achieve sufficient accuracy, thus requiring an extra processing stage. Methods that require full iDWT for feature extraction invalidate the major advantage of extracting feature information directly from compressed coefficients.

Research on feature extraction in compressed domain using SPIHT is very scarce whereas most of the related research has focused on compression applications (Section 1.5.2). In (Chang and Carin 2006), images are segmented prior to compression in order to rearrange the bit stream accordingly. The aim is to be able to quickly extract information about the segmented areas from the compressed image. The system suffers from the shortcomings of combined compression and feature extraction systems discussed earlier in this section. Furthermore, these schemes are not compatible with hardware implementations of the SPIHT algorithm such as (Nandi and Banakar 2008), which would enable easier adoption in practical applications.

SPEED OF SIMILARITY DISTANCE CALCULATION

Research towards more efficient distance calculation is scarce. Existing works include techniques such as splitting of the signature search space to reduce the processed data and manipulation of the bit-stream for progressive refinement of the distance calculation during decompression (Section 2.8). It must be noted that a common shortcoming of these approaches is that their performance is non-deterministic and highly source dependent (*i.e.* system response varies greatly with different input images). Consequently the publicly available results should be treated carefully when used as a point of reference for benchmarking other systems. A safer approach for comparison would be to re-implement those systems and compare responses both on a common dataset and with the same stimuli as the benchmarked system.



▲(a) (b)▼ (c) ▲

Figure 2.3: Number of articles retrieved by Google Scholar http://scholar.google.com

circa March '08

(a) Comparison between the '90s and the '00s for keyword sets *"texture+image+retrieval", "wavelet+image+retrieval", "texture+wavelet+image+retrieval"*

(b) Five-year breakdown of per-year incidents for keyword set *"texture+image+retrieval"* normalized over total incidents of keyword sets *"texture+image+retrieval"* and *"wavelets+image+retrieval"*

(c) Five-year breakdown of per-year incidents for keyword sets *"texture+image+retrieval", "wavelet+image+retrieval"* normalized over total incidents of *"image+retrieval"*.

2.11 Wavelet-based texture analysis in image retrieval research landscape

Recent reports (Datta, Joshi *et al.* 2008) indicate exponential growth of published works in the research area of Image Retrieval from 1995 to 2005 and a strong growth pattern thereafter. To investigate the validity of this assertion in the context of texture-based image retrieval in wavelets domain the following exercise was conducted. A series of queries were conducted on Google Scholar search engine (Google 2008b) using the key-phrases "texture", "wavelets" and "image retrieval" (Figure 2.3) and a series of observations were made based on the retrieved results.

Google Scholar is an aggregated index of peer-reviewed papers, theses, books, abstracts and articles, from academic publishers, professional societies, preprint repositories, universities and other scholarly organizations (including IET, IEEE, Elsevier, ACM and Springer). The graph of Figure 2.3(a) shows the number of

retrieved articles for three keyword sets: *texture image retrieval*, *wavelets image retrieval* and the combination *texture wavelets image retrieval*. The doubling of the references to *texture image retrieval* is to be expected given the general growth of interest in image retrieval. This two-fold growth of the references to *wavelets image retrieval* can also be justified since the wavelet-based image processing techniques and tools were still a niche field during the 1990s and started to mature during the 2000s.

An interesting observation is that, although references to *texture image retrieval* almost doubled, references to *texture wavelets image retrieval* were increased by approximately four times. This gives a clear indication of the strong linkage between wavelets and texture but also of the upward trend of new publications in this branch of image retrieval.

Figure 2.3(b) shows the references to *texture wavelets image retrieval* as a fraction of the broader key-phrases of *texture image retrieval* and *wavelets image retrieval* over the years 2003 to 2007. Observation of Figure 2.3(b) indicates that occurrences of key-phrase *texture wavelets image retrieval* account constantly for almost half of the references to the more generic key-phrase *wavelets image retrieval*, reinforcing the assertion of texture analysis being an important application of wavelets. More importantly, the graph of Figure 2.3(b) shows that from 2003 to 2007 there is an almost linear upward trend in *texture image retrieval* related references that encompass *wavelets*. Therefore, the two conclusions that can be drawn are: *a) Approximately half of the wavelets-based image retrieval articles are related to texture validating the strong linkage between them, b) There is a growing interest in texture-based image retrieval for wavelets domain techniques.*

Finally, from Figure 1.4(c) it becomes evident that for the last five years *the interest for texture and wavelets as a fraction of the overall interest in the research area of image retrieval has been steadily growing.*

2.12 Conclusions & problem statement

In conclusion, here is a summary of the shortcomings of the existing research identified from this literature survey. From this exposition of the published literature, it is possible to address a number of important issues, as follows:

- Major issues related to low-level CBIR performance have not yet been addressed effectively by the existing research. Accuracy and speed remain two of those core issues of image retrieval which require further investigation.
- The similarity between the characteristics of wavelets transformation and those of the HVS has made the former an excellent choice for texture analysis. Yet some pixel-based techniques offer better overall performance.
- Several practical applications of texture analysis are still in demand of efficient texture analysis techniques.
- Rotation invariance of wavelet-based texture descriptors is a "sensory gap" issue which requires further study as pixel domain counterparts seem to provide more efficient solutions. Furthermore, there is a clear gap when it comes to rotation invariant techniques which have the capability to be extended for direct extraction from wavelet-based compressed images.
- Quad-trees are at the heart of the wavelet transform and have been exploited for image compression and pixel-based feature extraction. However, they have not yet been applied as a mechanism for texture feature extraction from wavelet coefficients.
- A general issue of Image Processing systems, which also applies to image indexing and retrieval, is the need for methods that can efficiently extract feature information directly from the compressed bit-stream using standard compression schemes.
- Signature distance calculation is a significant contributing factor to the overall speed of an image retrieval query. Current research on improving the speed of distance calculation is limited, thus leaving a lot of scope for further studies.

It is evident that there is plenty of scope for improving wavelet-based feature extraction and image retrieval. Specifically, the focal point of this work is on the open

issues that exist in the fields of: wavelet-based texture characterization with emphasis on rotation invariance, retrieval speed improvement and on-the-fly feature extraction from compressed images.

Chapter 3: Texture Characterization using Wavelets

3.1 Abstract

In this chapter four techniques for feature extraction in wavelets domain are presented. These techniques offer three different perspectives to the utilization of the spatial information that is conveyed by the wavelet decomposition tree.

3.2 Introduction

The average energy of a DWT subband includes information about the amount of detail in the given direction and resolution (Mallat 1989). To this end, several existing techniques use 1^{st} or 2^{nd} order subband coefficient statistics as a texture descriptor (see examples in Chapter 2). An alternative representation of the energy levels of the wavelet coefficients is the Significance Map. A Significance map is typically a binary map containing the locations of those wavelet coefficients that exceed a specific energy threshold. Significance map encoding has been central to many wavelet-based compression schemes as a way to achieve energy compaction. The concise representation of Significance Map also attracted the interest of the image retrieval research community (see Section 2.7 for examples). The Significance Map brings the additional advantage of extracting features in parallel to the quantization and decoder stages hence requiring less decompression than schemes requiring the full wavelet coefficient values.

In this chapter, the issues of improving the accuracy and efficiency of texture-based image retrieval are addressed. Four methods are presented which take into account the local properties and statistics of DWT subbands. The first technique, called Localization Grid, suggests a low-cost method to improve the discrimination ability of energy-based texture characterisation by taking into account the local energy distribution within a subband. The second and third techniques introduce texture operators built around the Quad-tree structure. The fourth technique, called the Local Wavelet Coefficient Pattern (LWCP) builds upon an autoregressive occurrence

model applied on the coefficients of the directional subbands. The techniques are based on Significance Map Encoding offering compatibility with and easier embedding into compression codecs.

3.3 DWT and Significance Map Encoding

The concept behind Significance Map Encoding (SME) for compression applications is simply to disregard the least significant bits of the wavelet coefficients. The objective is to keep the minimum amount of information required for the reconstruction of the original image at an acceptable visual quality level (actually the aim in compression is usually the opposite *i.e.* to achieve better PSNR for a given bit rate). Shapiro elaborates on SME in (Shapiro 1993). Here follows a brief overview.

As discussed in Section 1.5.2, applying the wavelet transform to an image results in a low resolution image and a series of detail images (subbands). The low resolution image is obtained by iterative blurring of the original, whereas the detail images are the information that is lost during the operation. The DWT for the nth layer of decomposition is calculated by applying a separable filter bank, as follows:

$$LL_n(x,y) = \left[H * \left[H * LL_{n-1}\right] \downarrow 2,1\right]_{\downarrow 1,2}(x,y) \qquad \textbf{[eq 3-1]}$$

$$LH_n(x,y) = \left[H * \left[G * LL_{n-1}\right] \downarrow 2,1\right]_{\downarrow 1,2}(x,y) \qquad \textbf{[eq 3-2]}$$

$$HL_n(x,y) = \left[G * \left[H * LL_{n-1}\right] \downarrow 2,1\right]_{\downarrow 1,2}(x,y) \qquad \textbf{[eq 3-3]}$$

$$HH_n(x,y) = \left[G * \left[G * LL_{n-1}\right] \downarrow 2,1\right]_{\downarrow 1,2}(x,y) \qquad \textbf{[eq 3-4]}$$

where * is the convolution operator, $\downarrow$ 2,1 and $\downarrow$ 1,2 are the sub-sampling along the rows and columns respectively, LL_0 is the original image, H and G are low and band-pass filters respectively and x,y the coordinates of the target coefficient. LL_n is obtained by low pass filtering resulting in a low resolution image. The detail images (LH_n, HL_n *and* HH_n) are obtained via band-pass filtering. The original image is therefore represented by a set of sub-images (subbands) at several scales, in other words a multi-scale representation.

Let $c_0, c_1, \ldots, c_n$ be the wavelet coefficients of one of the four subbands *LL, LH, HL, HH*. Then, T_l is a series of thresholds described by:

$$T_0 = \frac{1}{2}\max\left\{\left|c_j\right| \mid j = 1,\ldots,n\right\} \text{ and, } T_l = \frac{T_{l-1}}{2} \quad l=1,2,\ldots,L \qquad \textbf{[eq 3-5]}$$

where L is the last threshold level which is determined dynamically when a predefined percentage of coefficients have been classified as significant according to the application. A wavelet coefficient is significant if its magnitude $|c_n|$ is greater than T_l. A Significance Map $SM(x,y)$ is therefore a binary map of the significant coefficients, that is:

$$SM(x,y) = \begin{cases} 1, & \text{coefficient at coords x,y is significant} \\ 0, & \text{coefficient at coords x,y is not significant} \end{cases} \quad \textbf{[eq 3-6]}$$

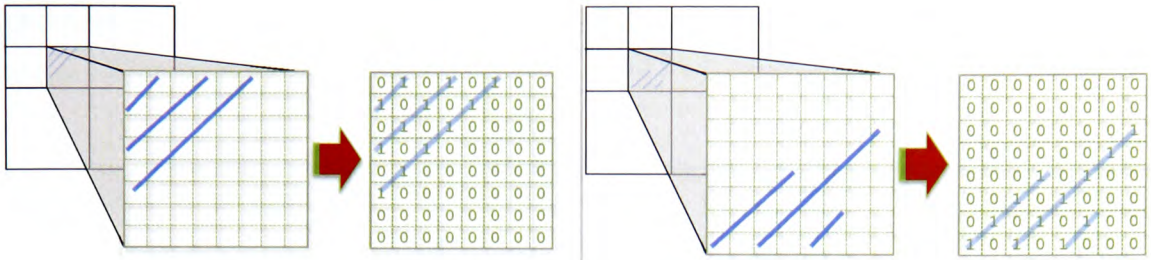
In texture analysis the Significance map offers a compact albeit incomplete picture of the energy distribution within a subband. To ensure that sufficient energy information for texture extraction is included algorithms typically refine the Significance Map using different thresholds iteratively.

A tri-state version of SME which includes significance information about the descendants (at the following decomposition level) of a wavelet coefficient is utilized by the third algorithm and is described in Section 3.4.3.

3.4 Algorithmic analysis

3.4.1 Localization Grid

It has been shown that sufficient information for the characterization of texture can be extracted via the relative importance of subbands (Liang and Kuo 1999). The importance of a subband is expressed in terms of the number of significant coefficients as it is defined in SME. Comparing the importance of subbands is analogous to comparing their energy content. From a texture similarity perspective if two images have similar importance (total number of significant coefficients for a given threshold) and similar distribution of significant coefficients per subband, they are classified as identical. Using the subband importance as the only metric suggests that the directional information (vertical, horizontal, and diagonal) is taken into account but not its local distribution. The algorithm is unable to detect the difference (see Figure 3.1) if at a given resolution (in this context represented by a specific level of decomposition) the same amount of directional information is either dispersed or concentrated at different spatial positions within the image/subband.



(a) number of significant coefficients = 12 (b) number of significant coefficients = 12

Figure 3.1: Example of mismatch when only global subband statistics are taken into account by the feature extraction algorithm.
(In reality the number of significant coefficients of each example might vary since the two shapes depicted have clearly different analogies of directional details. For the sake of the argument it is assumed that the two shapes generate equal proportion of horizontal, vertical and diagonal details.)

Existing texture analysis approaches based on the Significance Map include (Liang and Kuo 1999; Wilson and Bayoumi 2003; Au, Law *et al.* 2007) which use the importance of the DWT subbands as a texture feature and (Mandal and Liu 2004) which uses the Significance Map of the LL subband directly as the texture operator.

To this end, a low-cost improvement over subband importance is proposed, which provides the system with knowledge regarding the distribution of the significant coefficients inside a subband. In order to achieve that, an $lg_x \times lg_y$ binary map is created for each subband when processing the query image. This map indicates the locations inside one subband, where the concentration of significant coefficients is high. We call this binary map the *Localization Grid map*.

Here follows a description of the algorithms that are used when the retrieval system processes the query image (Algorithm 3.1) and one image from the database (Algorithm 3.2). Algorithm 3.1 calculates the localization grid and the texture vector for the query image whereas algorithm 3.2 uses the same localization grid to calculate the texture vectors of the images inside the database. Therefore, a complete query procedure requires the execution of algorithm 3.1 only once and of algorithm 3.2 for each image inside the database. Note that although other strategies could be followed, the choice of calculating the localization grid only once for the query image was made in order to: a) avoid the extra processing cost of calculating the localization grid for every image in the database which would impact the speed of the query as the database size increases, and b) emphasize the local

characteristics of the query image by masking out from the comparison process areas with low significance.

Algorithm 3.1: Calculation of Localization Grid map and Texture Vector for the query image *Q*.

Let $LG_i(m)$ be the $lg_x \times lg_y$ localization grid map for the i^{th} subband, $SMT(m)$ the total number of significant coefficients inside the mth block of the map, and TVq_{LG} the texture vector of the query image.

For a threshold T_l calculate the Significance Map of the query image.

1. Calculate the Localization Grid map:

   For each subband :

   a) Divide the subband into k equally sized blocks so that $k = lg_x \times lg_y$

   b) Count the number of significant coefficients in each block individually.

   c) Calculate the average number of significant coefficients per block.

   d) Build a binary map as follows:

$$LG_i(m) = \begin{cases} 1, & for \quad SMT(m) \geq average \\ 0, & for \quad SMT(m) < average \end{cases} \quad , m = 1,...,k \qquad \textbf{[eq 3-7]}$$

   For convenience, we call *active* the blocks that are marked with 1 in the localization grid map (see Figure 3.2).

2. Calculate the texture vector for the query image:

   For each subband:

   a) Let $SMTA_i$, be the total number of the significant coefficients (significance) which are located inside the *active* blocks of the i^{th} subband.

   b) Calculate $SMTAq'_i$, the normalized significance of the i^{th} subband of the query image according to:

$$SMTAq'_i = \frac{SMTAq_i}{\sum_i SMTAq_i} \qquad \textbf{[eq 3-8]}$$

   c) The texture vector TVq_{LG} is then given by:

$$TVq_{LG} = \{SMTAq'_1, SMTAq'_2 ... SMTAq'_n\} \qquad \textbf{[eq 3-9]}$$

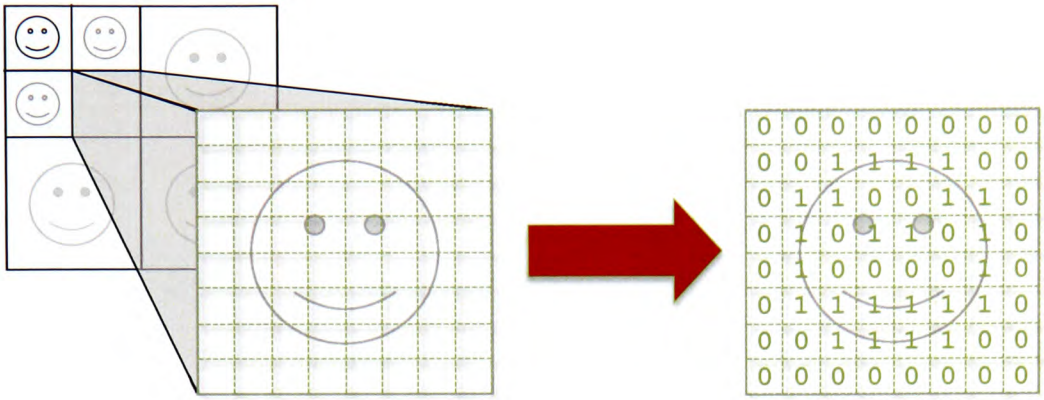


Figure 3.2: Example of an 8×8 localization grid for the HH subband of a DWT decomposition.
(Note that each active cell of the localization grid may aggregate more than one wavelet coefficient.)

Algorithm 3.2: Calculation of the Localization Grid Texture Vector for an image I inside the database

Let $LG_i(m)$ be the $lg_x \times lg_y$ localization grid map for i^{th} subband of the query image, and TV_{LG} the texture vector of image I.

Step 1: For a threshold T_l, dynamically calculate the Significance Map of image I.

Step 2: For each subband i :

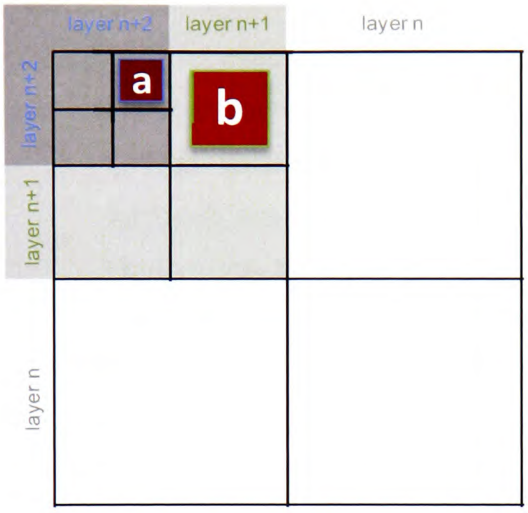
a) Calculate $SMTA_i$, the total number of the significant coefficients which are located inside the *active* blocks of the i^{th} subband of image I with respect to the localization grid map of the query image.

b) Calculate $SMTA'_i$, the normalized importance of the i^{th} subband of image I according to:

$$SMTA'_i = \frac{SMTA_i}{\sum_i SMTA_i} \qquad \textbf{[eq 3-10]}$$

The texture vector TV_{LG} is then given by:

$$TV_{LG} = \{SMTA'_1, SMTA'_2 \ldots SMTA'_n\} \qquad \textbf{[eq 3-11]}$$



Let S be the size (area) of the whole image. Then, S is also equal to the total size of the DWT decomposition. Each subband at any given level of decomposition has area equal to ¼ of the respective subband at the previous level of decomposition. Therefore:

Let b be the size of the 1st level of decomposition (*layer n+1*) then: $b = \frac{1}{4}S = \frac{1}{4^1}S$

Let a be the size of the 2nd level of decomposition (layer n+2) then: $a = \frac{1}{4}b = \frac{1}{16}S = \frac{1}{4^2}S$

Let w'_n be the size of the n^{th} level of decomposition then $w'_n = \frac{1}{4}w'_{n-1} = \frac{1}{4^n}S$

Figure 3.3: Derivation of size invariance weight for Localization Grid.

In order to compare two images with respect to their textures, the L_1-distance of their texture vectors is used:

$$d'_{L_1}(TVq_{LG}, TV_{LG}) = \left(\sum_i w'_i \cdot \left| SMTAq'_i - SMTA'_i \right| \right) \quad \textbf{[eq 3-12]}$$

where $SMTAq'_i$ and $SMTA'_i$ are the normalised significances of the i^{th} subband of the images to be compared and w'_i is the weighting factor described below.

Subbands at different levels of decomposition have different sizes, thereby affecting the value of subband significance since each coefficient contributes equally during the calculation. In order to compensate for this in eq.3-12, a weight is used, which is normalized with respect to the total size of the image (see Figure 3.3 for the derivation of this weight).That is, for the n^{th} level of DWT decomposition the weight it is given by:

$$w'_n = \frac{1}{4^n} \quad \textbf{[eq 3-13]}$$

3.4.1.1 Determination of the Localization Grid Size

A critical parameter of the algorithm is the dimensions of the localization grid maps. This must be determined and stored during the query image processing stage. The same maps are then used for the calculation of each image signature that is stored in the database. Localization Grid size affects: processing time, signature size and effectiveness of the algorithm. In particular, the size of the signature (the indexing

key) is affected by both the resolution of the localization grid as well as the number of subbands to which it is applied. The main tradeoffs when selecting the localization grid size and granularity are pointed out, as follows:

- A finer localization grid with size that is close to that of the respective subband would create a signature closely resembling the Significance Map. Using such a fine localization grid has the following disadvantages: a) the signature is very big, for example, a localization grid of 64×64 yields a binary map with 4096 bits or 512 bytes per subband b) the algorithm becomes very selective, losing its texture discrimination ability, thus selecting only images that have very similar Significance Maps. This increased selectivity does not reflect an increase in accuracy since the algorithm would show preference to images that are exactly like the query image rather than images with similar visual content. Images with small perceptual differences (*e.g.* a slightly shifted version of the same image) would produce very different signatures therefore characterized as very dissimilar by the algorithm;
- A very coarse localization grid with only a few cells will reduce the effectiveness of the algorithm in terms of spatial selectivity. Therefore, the improvement in accuracy over an algorithm that performs global per-subband statistical analysis will be cancelled.

The size of the image and the number of levels of decomposition must be taken into consideration when determining the size of the localization grid. The former is directly proportional to the size of the subbands and the latter defines the size of the subbands at the lowest level of decomposition. When selecting the size of the localization grid, the desirable target is therefore to achieve a good balance between granularity (cells small enough to provide sufficient spatial selectivity) and content capacity (cells large enough to avoid spatial over-selectivity as described above). In order to ensure that the Localization Grid remains within the above mentioned valid limits, a set of assertions were experimentally established, as follows:

- the smallest acceptable cell size is 4 x 4 coefficients *(content capacity)*;
- the smallest acceptable localization grid size is 4 x 4 cells *(granularity)*;
- the maximum number of decomposition levels to which the localization grid is calculated is 4 *(limiting signature size)*.

The smallest applicable subband size can therefore be 16×16 coefficients (*i.e.* a Localization Grid 4×4). The first two assertions are consequently valid for images larger than 32×32 pixels since a one-level decomposition would lead to a subband size of a subband size of 16×16 coefficients. Practically this limit does not reduce the usefulness of the algorithm since image resolution 32×32 is already far too small sample to perform meaningful statistical analysis for feature extraction. The third assertion was set in order to impose a hard limit to the signature size, thereby constraining the overall processing cost. Experiments on how different Localization Grid sizes affect accuracy are presented and the above mentioned assertions are validated in Section 3.6.1.

3.4.1.2 Calculation of threshold

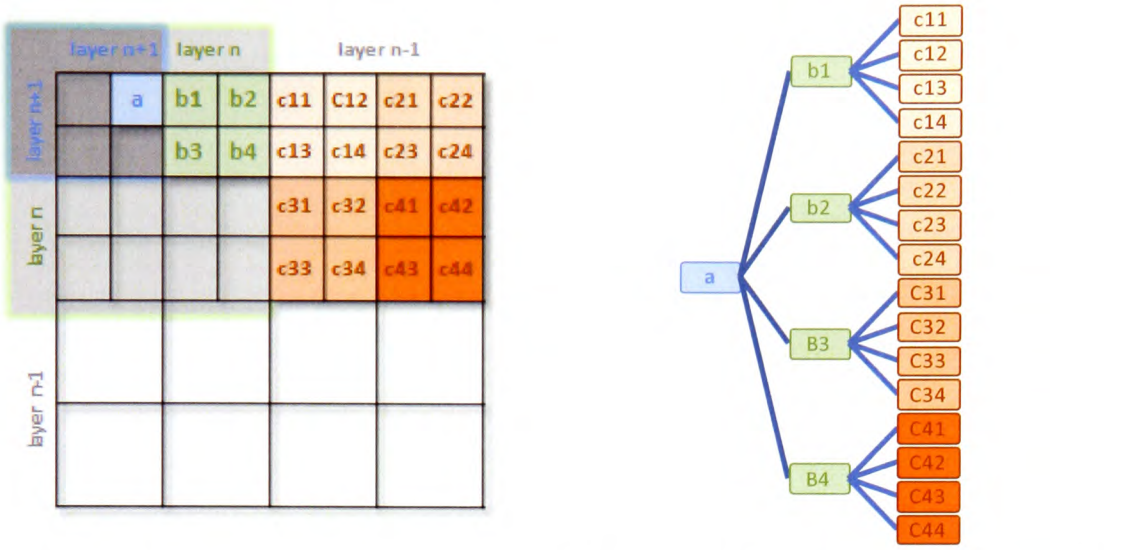
The calculation of the threshold used during the Significance Map creation follows the principles of the quantization stage implemented by many wavelet-based codecs (Shapiro 1993; Said and Pearlman 1996b; Liang and Kuo 1999). Two different strategies are followed depending on whether the processed images are compressed or not, as follows:

Uncompressed Images

1. At a given decomposition level, starting from a threshold equal to:
   $threshold = (max\ coefficient\ value - min\ coefficient\ value)/2 + min\ coefficient\ value$
2. Lower the threshold until the number of significant coefficients at all subbands equals $1/3$ of the size of the subbands or until the 7^{th} order of magnitude is reached (the threshold is lower than 1% of the maximum at that point). This scheme has been found to provide sufficient information for texture extraction whilst keeping the noise artefacts to low levels in (Liang and Kuo 1999; Au, Law *et al.* 2007)
3. Record this value and store along with the signature.

Compressed Images

Significance map encoding is typically employed by most wavelet-based compression algorithms. The threshold is defined during encoding and is commonly calculated using a similar strategy to the one described above for the uncompressed images. It is therefore convenient and desirable for savings in processing to take into consideration the predefined thresholds for feature extraction. This issue is discussed



(a) A DWT decomposition

(b) Equivalent Quad-tree representation of the denoted coefficients

Figure 3.4: Analogy between Quad-trees and the DWT tree

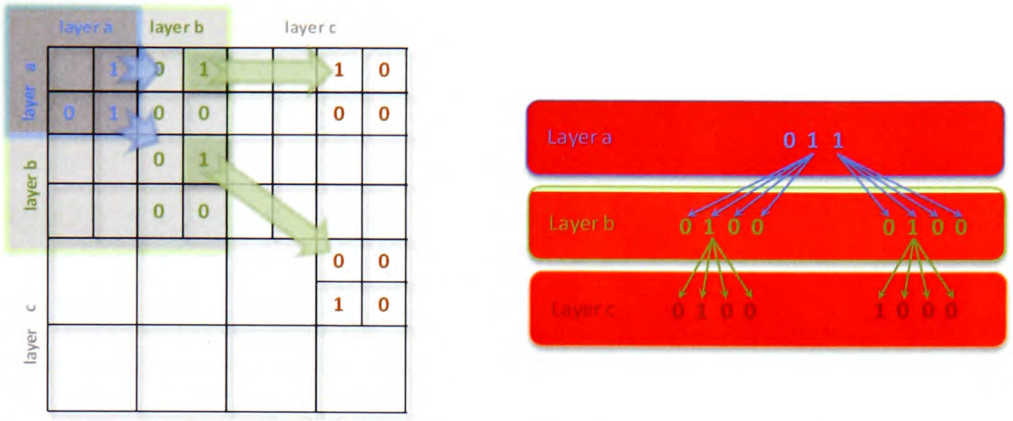
in more detail in Chapter 5 where a feature extraction technique in native compressed domain is presented.

3.4.2 Quad-trees

Discrete wavelet decomposition uses a Quad-tree structure (Shapiro 1993) to perform partitioning of the DWT coefficients. This is depicted in Figure 3.4. As it can be seen, each root node has four offspring (in analogy to Quad-tree leaves), in the adjacently higher level of decomposition.

In the proposed architecture the DWT coefficients are not used directly as the source data for building the quad-tree. The Significance Map is used instead, which inherits the structure of the DWT tree, thereby reducing the data processing requirements. Figure 3.5a shows an example of a Significance Map and Figure 3.5b its quad-tree representation.

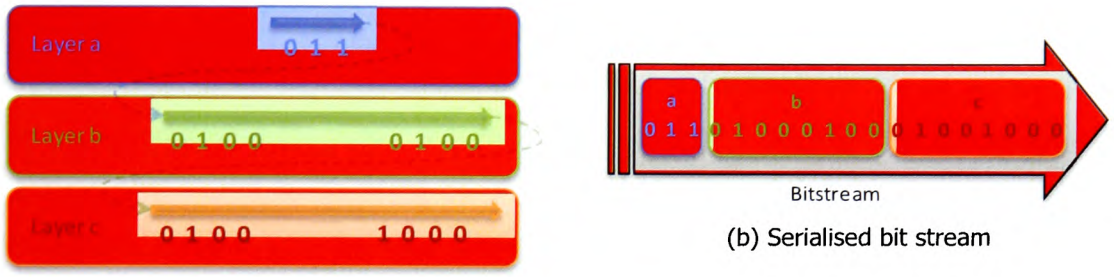
An *internal node* is a node of the quad-tree structure which has descendants (*e.g.* nodes denoted by 1 in Figure 3.5b). A *leaf node* is the node of the quad-tree structure which has no descendants (*e.g.* nodes denoted by 0 in Figure 3.5b), respectively. The signature for one image is then constructed as follows:



(a) Partial 8×8 Significance Map. (blanks ignored)

(b) corresponding Quad-tree structure

Figure 3.5: Analogy between Significance map and Quad-tree structure



(a) encoding of Quad-tree structure

(b) Serialised bit stream

Figure 3.6: Quad-tree indexing signature encoding

Starting from the upper-most level (LL subband) each leaf node is encoded with 0 and each internal node with 1 (see Figures 3.6a & 3.6b). The outcome of this process is a bit-series which forms the feature vector for the specific image.

The similarity between two images is then calculated by XOR-ing their Quad-tree vectors. The procedures for generating a quad-tree vector and comparing two images are described in more detail in Algorithms 3.3 and 3.4 respectively.

Algorithm 3.3: Creation of Quad-tree signature

For each image inside database, for a given threshold T_l:

1. Calculate the DWT tree.
2. Calculate the Significance Map for a given threshold T_l

3. For each significant coefficient in the LL subband build its quad-tree structure using the following rules:
   a) If any of the descendants of the coefficient are significant then output 1 (internal node)
   b) If none of the descendants of the coefficient is significant then output 0 (leaf node)

Repeat steps 3(a) and 3(b) recursively for each of the designated internal nodes.

Algorithm 3.4: Comparison of two images using Quad-tree indexing

Let V_q be the bit vector produced for the query image and V_t the bit vector produced for an arbitrary image in the database, then:

For each of the "root" nodes (those that were found significant in the LL subband) of Image Q: Compare with every "root" node of Image T by XOR-ing the respective vectors V_q and V_t and by counting the number of *1*s produced.
The number of *1*s is the distance between the two images. A higher number of *1*s means a smaller relevance and vice versa.

The choice of T_l is very important since it directly affects the available content information about the image, as well as the size of the produced quad-tree. The methodology of Section 3.4.1.2 is used.

Another desirable feature is to have a small yet accurate key. As more decomposition layers are added to the calculation of the Quad-tree, the structure and the signature becomes larger. Theoretically, the only upper limit is the total size of all subbands that are considered during calculation, in the hypothetical case that all coefficients are significant. This leads to high uncertainty and unpredictability of the signature size before calculation and potentially a cumbersome signature. In practical applications it is desirable to have prior knowledge of the signature size for storage and processing requirements planning.

In order to avoid saturation of the data sample and considering also the fact that the energy in the DWT decomposition tree is concentrated in higher thresholds and in

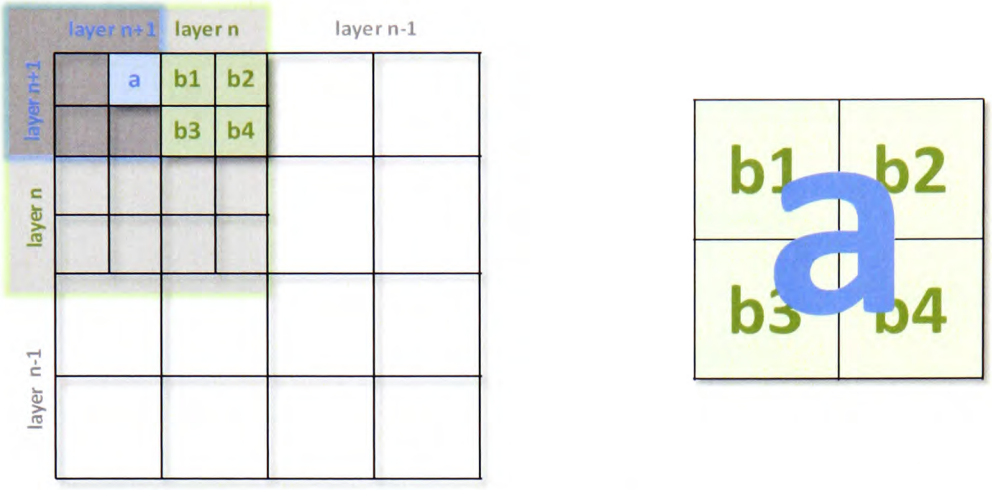
the lower subbands (Mallat 1989; Liang and Kuo 1999) a threshold ensuring that a maximum of 30% of the coefficients will be characterized as significant is used to constrain the required processing load during the query. The threshold was established empirically in order to preserve sufficient content information for indexing. Studies on the depth of the Quad-tree signature and the location of the root nodes are presented in Section 3.6.

The operator of the Quad-tree algorithm aggregates information by several layers of decomposition in a single feature signature. This characteristic offers the advantage of correlating information between different resolutions. Also, due to the top-down coarse-detail approach it embeds regional information about texture thereby approaching the behaviour of a shape descriptor. The trade off of this design is that images with distinct textured regions and global features will be favoured over images with finer and sparser texture. Additionally the comparison of two vectors by XOR-ing is sensitive to affine transformations since these would affect the spatial distribution of the significant coefficients hence the structure of the feature vector. Affine transformations include linear transformations (rotation, scaling or shear) and translations (or "shifts"). The issues identified here are addressed in an enhanced version of the Quad-trees algorithm called Quad-tree Patterns, which is presented in the next Section.

3.4.3 Quad-tree Patterns

The concept of quad-trees is used in this section as a basis in order to construct another texture descriptor. Instead of looking at the wavelet decomposition from a macroscopic point of view, the Quad-tree representation is now used on micro and inter-subband characteristics of the DWT tree.

In the previous section, the Quad-tree characteristics of the wavelet decomposition were established. Starting from a coefficient at a lower frequency subband, there are four descendants at the respective position of the immediately higher frequency subband. When a natural image is transformed using wavelets most of the energy tends to gather at the lower energy subbands (Mallat 1989; Liang and Kuo 1999). In other words, coefficients at lower frequency subbands tend to have greater magnitudes than those in higher subbands. In physical terms this means that smooth



a) three consecutive layers of DWT decomposition

b) superimposed coefficients at two consecutive levels of decomposition

Figure 3.7: Formation of the Quad-tree Pattern signature – part A

(coarser) areas tend to dominate over finer details and highly textured areas in natural images (Ghanbari 2003).

Consequently, it is expected that during Significance Map encoding the coefficients belonging at lower frequency subbands will have higher significance and appear earlier in the bit-stream than those in higher frequency subbands. Also, the probability of having an insignificant root node with significant descendants, for a given threshold, is very low.

The main feature of EZW is that during Significance Map encoding each coefficient is encoded using a tri-state symbol. The first two states indicate whether the coefficient is significant or insignificant, and the third state indicates that the coefficient is insignificant and all its immediate descendants are insignificant too, therefore it is marked as a leaf node.

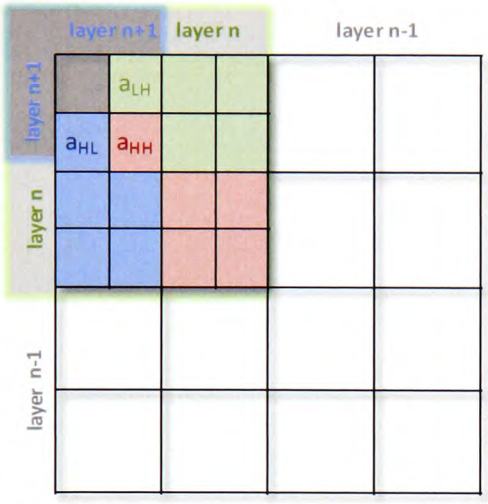
This paradigm is used in order to construct a Quad-tree-based texture operator using the information obtained by two consecutive levels of decomposition. One root node and its four immediate descendants are encoded using the tri-state symbolization described above (see Figure 3.7).

Let a be a wavelet coefficient at decomposition layer $n+1$ and b_1, b_2, b_3, b_4 the immediate descendants, as shown in Figure 3.7. The significance status ss_a of coefficient a, as defined by EZW encoding, is given by the following formula:

$$ss_a = \begin{cases} 0, & a \text{ is insignificant and none of } b_1,\ldots,b_4 \text{ is significant} \\ 1, & a \text{ is significant} \\ 2, & a \text{ is insignificant and one of } b_1,\ldots,b_4 \text{ is significant} \end{cases} \quad \textbf{[eq 3-14]}$$

Each consecutive layer of decomposition represents a down-sampled version of the previous layer. As it is shown in Figure 3.7(b), topologically, one "parent" coefficient represents the same area covered by the four descendants at the previous layer of decomposition. Therefore, associating the information of two consecutive decomposition layers leads to correlation between the lower frequency information of the "parent" coefficient and the higher frequency information of the descendants. Research of the human vision system (HVS) shows that, the wavelet-based spatial-frequency representation can preserve efficiently both global and local information, and is suitable for modelling quasi-periodic signals. As aforementioned, it has demonstrated a remarkable performance for texture classification and analysis (Tuceryan and Jain 1993; Mandal, Idris *et al.* 1999). Correlating the coefficients of only two consecutive decomposition layers has the potential of a feature design that incorporates texture discrimination and multi-resolution properties.

Limiting the pattern to only one "parent coefficient" and two consecutive layers of decomposition makes the algorithm more capable to detect finer textures patterns than the original Quad-trees. This is justified by the fact that the leaf nodes (see Figure 3.5) of the original Quad-tree algorithm are always attached to a parent node at the LL layer which corresponds to coarse information, *i.e.* a larger section of the original image. Due to this bound relationship the algorithm cannot isolate and therefore detect smaller patterns that could possible appear within the branches (i.e. detail subband information) of the same parent node. Here, the two-layer version of the operator can be applied to n-1 layers of decomposition; where n is the total number of layers of decomposition of an image. Thus, the proposed operator has the



a) Three directional coefficients that are used for the construction of the descriptor. The areas highlighted with the same colour designate the descendants that determine the state of the parent.

LH	HL	HH
0	0	0
0	0	1
0	0	2
0	1	0
0	1	1
0	1	2
0	2	0
0	2	1
0	2	2

LH	HL	HH
1	0	0
1	0	1
1	0	2
1	1	0
1	1	1
1	1	2
1	2	0
1	2	1
1	2	2

LH	HL	HH
2	0	0
2	0	1
2	0	2
2	1	0
2	1	1
2	1	2
2	2	0
2	2	1
2	2	2

b) The 27 different states that define the texture descriptor, as defined in eq.3-14 (0=insignificant with insignificant descendants, 1=significant, 2=insignificant with significant descendants)

Figure 3.8: Formation of the Quad-tree Pattern signature – part B

potential to equally capture finer and coarser information at all layers of decomposition.

Each parent coefficient corresponds to 4 descendants, which belong to one of the three directional subbands (As shown in Figures 3.5a and 3.7a). Starting from the last layer of decomposition 4 coefficients exist that represent a specific image location; one coarse and three directional. The coarse coefficient is not taken into consideration for the design of the texture descriptor. Thus, one coefficient per direction is left, for the construction of the descriptor. Each of the three coefficients has four descendants and can be described with one of the three aforementioned states. In order to incorporate multi-directional information of a specific image location all three directional coefficients are used simultaneously for the construction of the texture descriptor (see Figure 3.8a). For a given set of coordinates, the texture descriptor can have one of 27 states (see Figure 3.8b) *i.e.* a histogram of 27 bins (elements) can be created in order to register the occurrences of each state for one level of decomposition. In order to strengthen the discriminative ability of the algorithm, the normalized histograms of three levels of decomposition are used for

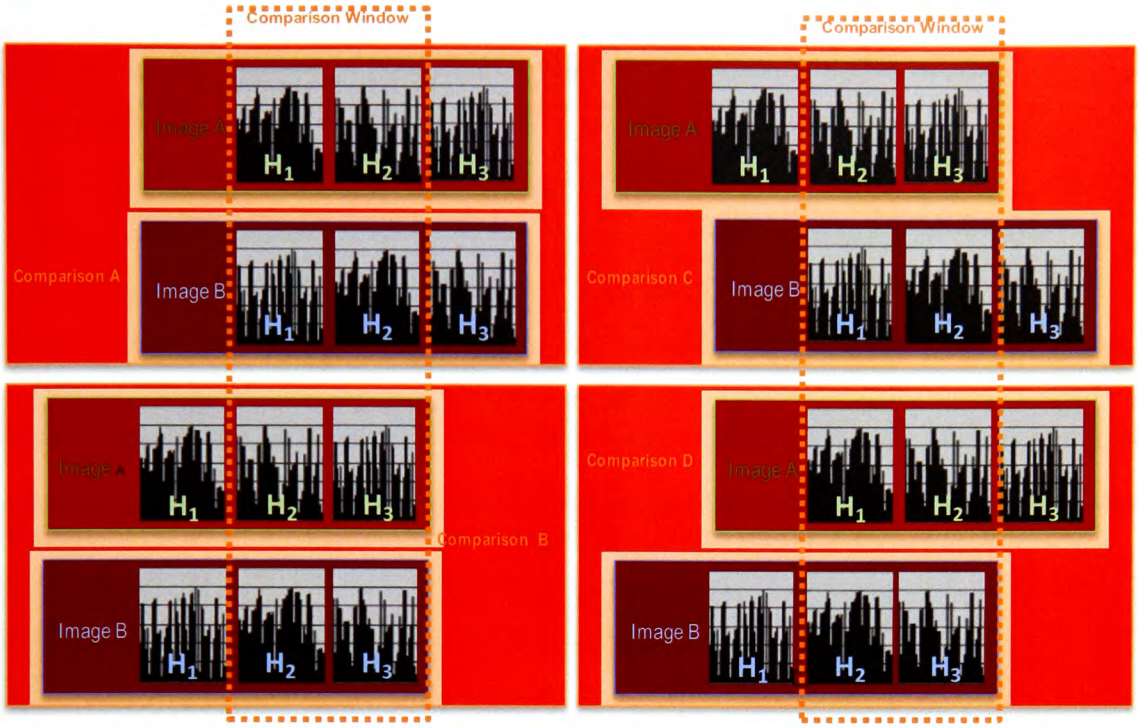


Figure 3.9: Methodology for multi-resolution comparison of two images using Quad-tree Pattern signatures.

the construction of the final version of the descriptor, thus creating a histogram of 81 bins.

Each set of histograms that is generated for a given level of decomposition, corresponds to a specific resolution scale. In order to enhance the discrimination ability of the algorithm, a form of scale invariance is implemented for dyadic scale differences following the architectural principles of the DWT decomposition. Dyadic scale differences are inherent in DWT since each decomposition level differs from the previous one by a factor of two. To achieve that, when comparing the signatures of two images, the histograms are compared against two consecutive histograms in all possible permutations *i.e.* the comparison is performed using a sliding window of two histograms (see Figure 3.9 and algorithm 3.6).

Algorithm 3.5: Calculation of Quad-tree Pattern Signature

For each image inside database, for a given threshold T_l:

1. Calculate the DWT tree.
2. Calculate the Significance Map for a given threshold T_l

Build the histogram for the three lower (higher frequency) levels of decomposition:

3. Let subband size at level L equal to $M \times N$ coefficients. Using eq.3-14 determine the state of the three coefficients at coordinates c_x, c_y in LH, HL and HH subbands for $c_x = 0,1,...,M$ and $c_y = 0,1,...,N$, thus construct a 27-bin histogram for level L
4. Repeat step 3 for the two consecutive levels of decomposition
5. Normalize and concatenate the three 27-bin histograms to one 81 bin histogram, which is the final form of the texture signature

Algorithm 3.6: Comparison of two Quad-tree Pattern Signatures

Let H_a1 , H_a2 , H_a3 be the histograms of the three levels of decomposition for image ***a*** and H_b1 , H_b2 , H_b3 the histograms for ***b*** respectively.

Step 1: H_b1 H_b2 is compared to H_a1 H_a2

Step 2: H_b1 H_b2 is compared to H_a2 H_a3

Step 3: H_b2 H_b3 is compared to H_a1 H_a2

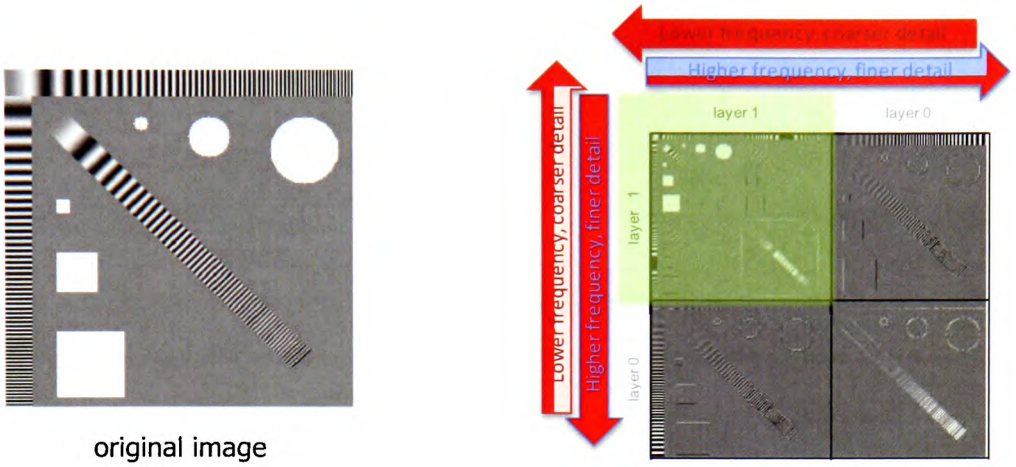
Step 4: H_b2 H_b3 is compared to H_a2 H_a3

The comparison of each step is performed using the L1-distance of the respective Quad-tree Pattern histograms and the methodology of figure 3.9 to achieve scale-invariance. The best match of the four comparisons is used as the final distance between the two images.

3.4.4 Local Wavelet Coefficient Patterns

As it was discussed in Chapter 2, autoregressive occurrence models have been successfully used for texture characterization in pixel domain. One of the most prominent examples is (Ojala and Pietikainen 2002). The autoregressive occurrence model suggested by Ojala is the Local Binary Patterns (LBP). The LBP model detects circular patterns with respect to a central reference pixel, which are formed by different textures in an image.

Images that have been decomposed using the wavelet transform contain subbands or sub-images that convey information regarding the directional details at a given



original image

two-level wavelet decomposition

Figure 3.10: A two-level Discrete Wavelet Transform decomposition.

resolution. For quadrature decomposition, the directional information includes horizontal, vertical and diagonal details. Higher frequency subbands depict finer details or, in other words, information about sharper edges in the image. Lower frequency subbands have information about coarser details or smoother areas of the image (see Figure 3.10). Thus, the latter represents changes at greater sections in an image, that is, one wavelet coefficient represents a greater number of pixels than a coefficient at a higher frequency subband. From a different perspective, these lower subband coefficients represent less sudden changes in the texture of the image or less sharp edges.

The LWCP method uses an autoregressive occurrence model based on the fundamental principle of circular patterns in order to exploit local characteristics of the image which are contained in wavelet-domain subband coefficients. LWCP analyses edge patterns in DWT in order to extract texture information. In order to provide a better understanding of the circular pattern model, as this is applied in pixel domain, this section begins with a brief description of the basic LBP operator.

Figure 3.11 depicts the basic $LBP_{8,1}$ operator. The subscript 8 refers to the number of surrounding pixels and the subscript 1 refers to the radius of each pixel from the centre. The $LBP_{8,1}$ operator detects the edges surrounding a central pixel at 45° intervals by subtracting the grey value of the centre pixel (g_0) from the grey values of

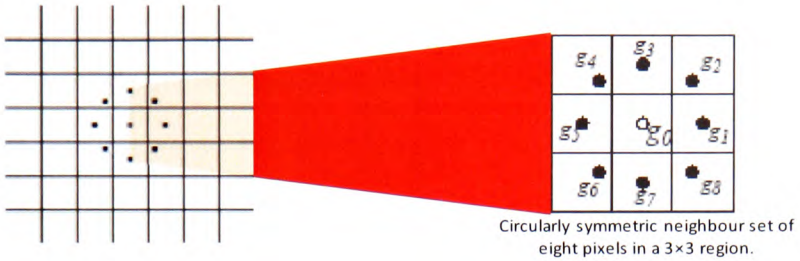


Figure 3.11: The Local Binary Pattern LBP8,1 operator (Ojala, Pietikainen et al. 2002b)

the eight surrounding pixels (g_i , $i=1,...,8$). The operator is then constructed by considering just the signs of the respective subtractions, *i.e.*:

$$LBP_{8,1} = s(g_0 - g_c), s(g_1 - g_c), ..., s(g_7 - g_c) \qquad \textbf{[eq 3-15]}$$

where,

$$s(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases} \qquad \textbf{[eq 3-16]}$$

$$LBP_{8,1} = \sum_{p=0}^{7} s(g_p - g_0) 2^p \qquad \textbf{[eq 3-17]}$$

In wavelet decomposition edges correspond to the energy represented by the magnitude of the coefficients that constitute each subband. Horizontal, vertical and diagonal edges correspond to coefficients belonging to the respective *HL, LH* or *HH* subband of each level of decomposition. In pixel domain, the LBP operator correlates the information about intensity variations surrounding a specific spot in an image. In order to draw an analogy in the wavelet domain, the LWCP algorithm extracts edge information about a specific location in an image by correlating the coefficients that appear at specific coordinates collectively in all *HL, LH* and *HH* subbands, at a specific level of decomposition. Since the subband coefficients represent edges, LWCP detects patterns of neighbouring edges. Let $c_{x,y}$ be a DWT coefficient at coordinates x,y , then for each of the *HL, LH* and *HH* subbands, the following differences are first calculated (see Figure 3.12):

$$C^{LH}_{x-1,y} - C^{LH}_{x,y} \qquad C^{LH}_{x+2,y} - C^{LH}_{x,y}$$

$$C^{HL}_{x,y-1} - C^{HL}_{x,y} \qquad C^{HL}_{x,y+1} - C^{HL}_{x,y}$$

$$C^{HH}_{x+1,y+1} - C^{HH}_{x,y} \qquad C^{HH}_{x-1,y-1} - C^{HH}_{x,y}$$

$$C^{HH}_{x-1,y+1} - C^{HH}_{x,y} \qquad C^{HH}_{x+1,y-1} - C^{HH}_{x,y}$$

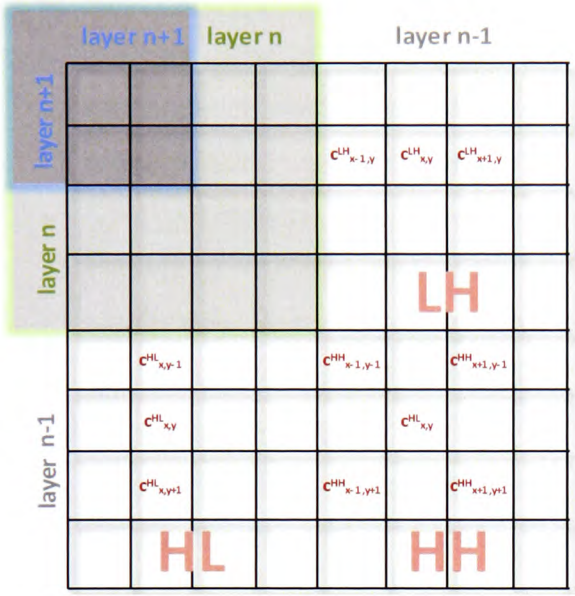


Figure 3.12: Derivation of the LWCP operator

The operator is then constructed by considering just the signs of the respective subtractions. Following the paradigm of eq.3-17 the LWCP operator takes the form of an eight-bit long binary mask as follows:

$$LWCP = s(c^{LH}_{x-1,y} - c^{LH}_{z,y}) + s(c^{LH}_{x+1,y} - c^{LH}_{z,y})2 + s(c^{HL}_{x,y-1} - c^{HL}_{z,y})2^2 + s(c^{HL}_{x,y+1} - c^{HL}_{z,y})2^3 + \\ + s(c^{HH}_{x+1,y+1} - c^{HH}_{z,y})2^4 + s(c^{HH}_{x-1,y-1} - c^{HH}_{z,y})2^5 + s(c^{HH}_{x-1,y+1} - c^{HH}_{z,y})2^6 + s(c^{HH}_{x+1,y-1} - c^{HH}_{z,y})2^7 \quad \textbf{[eq 3-18]}$$

where, each bit corresponds to the sign of each of the subtractions.

This algorithm operates directly on wavelet coefficients, thus it can be applied to any level of decomposition as long as the subband has a minimum size of 3×3 pixels. A higher level of decomposition has higher energy content and represents a coarser scale. A lower level of decomposition has lower energy content and represents a finer level of detail, containing a greater number of coefficients. The selection of the level of decomposition is therefore a trade-off, since the level of detail between different textures may vary considerably.

3.4.4.1 Boundary conditions

An issue that arises due to the finite 2-dimensional space of the DWT tree is the behaviour of the algorithm in boundary conditions, *i.e.* for the coefficients that are on the edge of the subband. These coefficients do not have a complete set of neighbours in order to calculate eq.3-18. The possible strategies are (Strang and Nguyen 1996; Tsui 2004):

- Zero-padding: it is assumed that the signal is zero outside the original support. The disadvantage of zero-padding is that discontinuities are artificially created at the border leading to visible artefacts after inverse-DWT and is usually unacceptable.
- Symmetrization: This method assumes that signals or images can be recovered outside their original support by symmetric boundary value replication. Symmetrization has the disadvantage of artificially creating discontinuities of the first derivative at the border, but this method works well in general for images.
- Smooth padding of n-order: This method assumes that signals or images can be recovered outside their original support by a simple n-order derivative extrapolation: padding using a linear extension fit to the first n- and last n-values. Smooth padding works well in general for smooth signals.
- Periodic-padding: This method assumes that signals or images can be recovered outside their original support by periodic extension. The disadvantage of periodic padding is that discontinuities are artificially created at the border.

The DWT associated with these four modes is inevitably redundant. But inverse-DWT (iDWT) ensures a perfect reconstruction for any of the four previous modes whatever the extension mode used for DWT.

These padding techniques are used in cases where convolution filters are applied to discrete finite signals (*e.g.* the DWT) (Strang and Nguyen 1996; Tsui 2004). As long as the LWCP algorithm scans for patterns in the detail subbands, *i.e.* the larger subbands of the DWT tree, it can be safely assumed that the exclusion of the boundary coefficients from the calculation can be accommodated without significant loss of information. Nevertheless, the aim of a texture extraction technique is to analyze the periodicity of specific image visual characteristics rather than to precisely reconstruct an image from the wavelet coefficients. For a 256×256 (65636 coefficients) subband if the boundary coefficients are omitted (256×4 – 2 = 1022), this represents a loss of approximately 1.5% of the available content information (wavelet coefficients) or 98.5% of the coefficients are used for the calculation of LWCP.

3.4.4.2 Signature structure

Two options are proposed regarding the formation of the signature:

1. Quantized Histogram: Calculate the full histogram as a signature. The size of the histogram depends on the size of the subband, for example, for a 256×256 coefficients subband each bin should hold up to 65Kbytes maximum, whereas for 256 bins that would be 2Mbytes per subband. After calculating the histogram the bins are quantized to 64 steps in order to reduce the size of the signature. The size of the key is equal to the number of bins multiplied by the levels of quantization, *i.e.* for 64 levels of quantization 7bits×256 = 224 bytes per subband.
2. Histogram Binary Map: Create a 256 bit binary map, each bit representing one bin of the histogram. Given a threshold T: assign to 1 each bit that represents a bin with a greater or equal to the number of hits than T and 0 each bit that represents a bin with a lower number of hits than T. This method greatly reduces the size of the signature down to 32 Bytes per subband. However a lot of the feature information is discarded therefore increasing the possibility of error. The threshold T should also be stored with the signature and should be used when calculating the feature vectors of other images for comparison. The threshold value can be different for each subband.

Summarizing here follows the generic LWCP algorithm:

Algorithm 3.7: Creation of LWCP signature

For a subband of size $(Sx \times Sy)$, where $Sx > 3$ and $Sy > 3$,

1. Create a histogram with 256 bins. Each bin corresponds to one of the possible 256 8-bit patterns.
2. For each coefficient with coordinates x,y calculate eq. 3-18, where $2 > x > Sx\text{-}1$ and $2 > y > Sy\text{-}1$ in order to disregard boundary coefficients.
3. Increment the value of the bin designated by the result of step 2
4. Use one of the techniques mentioned above for the creation of the signature, that is: Quantized Histogram or Histogram Binary Map.
5. The quantization map (for Quantized Histogram) or the threshold (for Histogram Binary Map) should be recorded respectively as they constitute part of the signature.

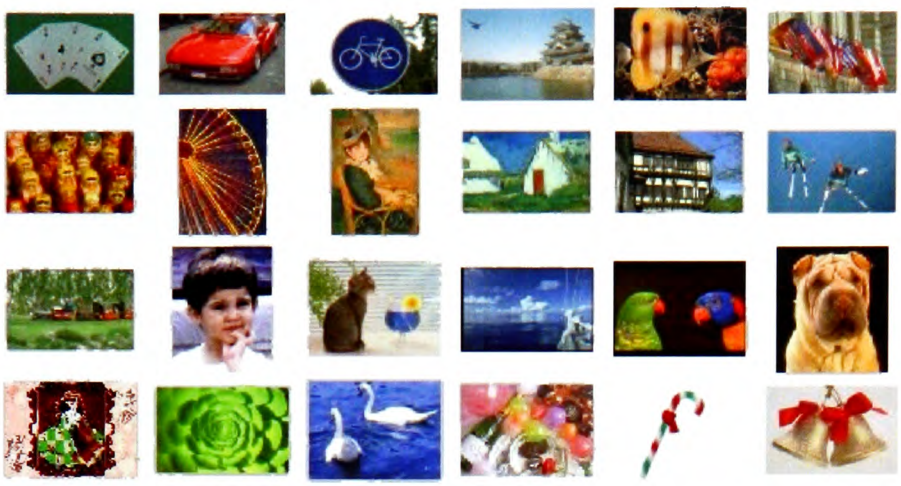


Figure 3.13: Database A; selected sample images.

3.5 Experimental setups

3.5.1 Datasets

Three databases have been used for the experiments; Database A contains natural and composite images of various contents, Database B various textures, and, Database C grouped textures according to their pattern perceptual characteristics.

3.5.1.1 Generic Image Indexing and retrieval (Database A)

The retrieval experiments were performed on an in-house compiled database of approximately 1000 images which is available at (Voulgaris 2008) with mean size 512×256 pixels. The image set included a plethora of diverse content. The content varies in terms of theme including landscapes, human faces, animals, aerial photographs *etc.* Consequently, there is variety in the dominant characteristic feature of the database images (*e.g.* large smooth areas in contrast to large areas with many edges). Figure 3.13 shows samples from the database. Note that in all experiments the images that were used as examples for querying were automatically excluded from the database. The grey scale versions of the images were obtained by transforming the original RGB values to the YUV colour space according to conversion table 3.1 which is based on the ITU recommendation (ITU-R 2007). The Y (Luminance) channel was then scaled to the range of 0-256 to obtain the grey-scale image.

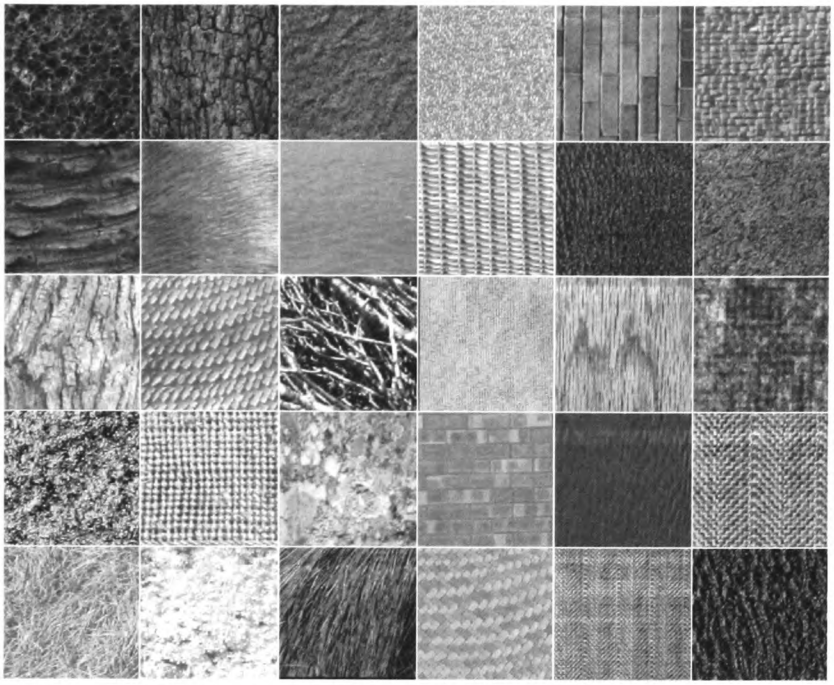


Figure 3.14: Database B; texture classes.

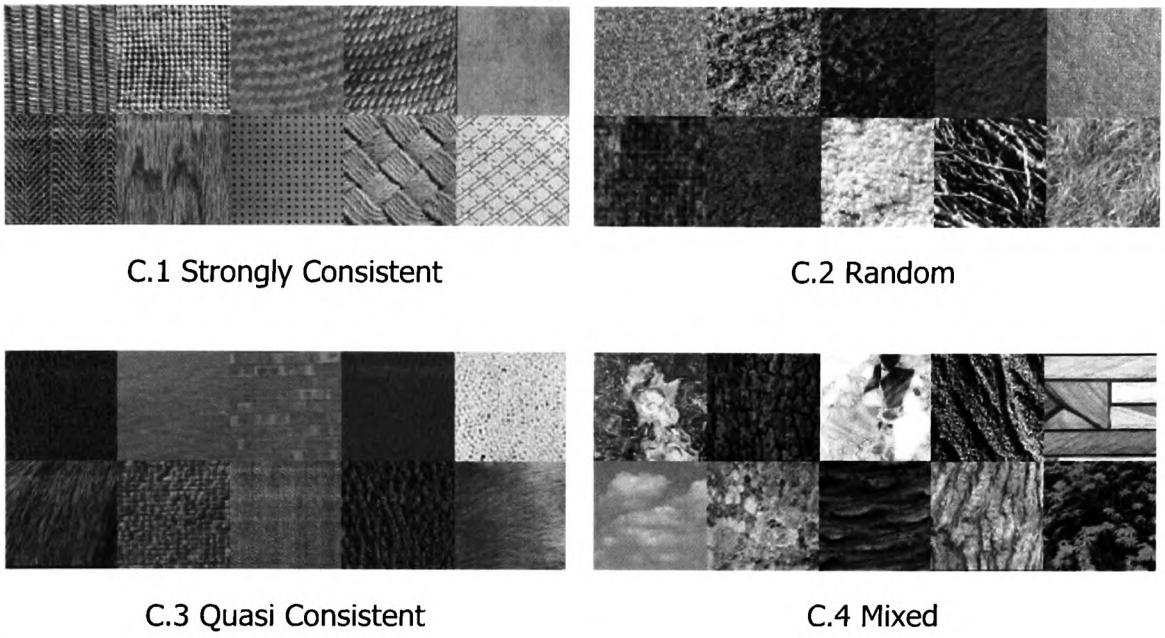
Table 3.1: Conversion of RGB to YUV colour space (ITU-R 2007)

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.1 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

3.5.1.2 Classification (Databases B & C)

Following the analysis of Section 1.6.4, a database of 1920 image regions extracted from 30 Brodatz texture classes was constructed by subdivision of 512×512 pixels texture images into 64 non-overlapping 64×64 pixel regions (Figure 3.14). The Brodatz image set was excerpted from the Outex Image Database of the University of Oulu, Finland (Ojala, Maenpaa et al. 2002).

In order to investigate the effectiveness of the algorithms with respect to the perceptual characteristics of the textures, the taxonomy of Figure 3.15 was used containing four distinct classes: C.1 strongly consistent patterns, C.2 random patterns, C.3 quasi consistent patterns, C.4 mixed texture patterns. The classes were defined with respect to the uniformity of the texture patterns globally, locally (i.e. how easily can textured regions be identified within an image when the texture is not globally uniform) and also on the whether the local (at any distinct region) or linear



C.1 Strongly Consistent C.2 Random

C.3 Quasi Consistent C.4 Mixed

Figure 3.15: Database C; Textures grouped based on perceptual characteristics

(along a virtual straight line) perceived periodicity of a texture pattern is regular or random.

3.5.2 Evaluation of features

3.5.2.1 Generic or Natural Image Indexing and Retrieval accuracy

As it was described in Sections 1.4 and 1.6.4, when the contents of an image database span a broad image domain, *i.e.* the semantic content of the images is unpredictably variable, it is very difficult to establish a ground truth with regards to image similarity. The reason is the differences in perceptual interpretation of a scene by different observers.

Therefore, the issue of perception variability makes it difficult to use the traditional metrics of *precision*, *recall* and their derivatives as they rely on well defined distinctions between similar and dissimilar images. In order to compensate for this variability researchers such as (Fahmy, Black et al. 2006; Teynor, Muller et al. 2006a) incorporated user surveys into their studies, with the aim of validating the performance of their systems.

Here database A was used in order to assess the effectiveness of the algorithms for retrieval from a very diverse dataset. In order to provide more meaningful results ground truth was established by grouping and labelling semantically similar images by four members of the digital imaging research group so that the *precision* metric can be used to quantify accuracy.

3.5.2.2 Classification accuracy

The performance of a classification system does not only depend on the efficiency of the feature characterization method. It is also directly affected by the statistical analysis technique of the extracted features which in this context is the classifier. This section describes some of the most commonly used practices for classification and for evaluating classification accuracy. The section concludes with the choices that were made for this work.

Let an instance q (the query) have attributes $\vec{q}_v$ (the feature vector). The K nearest neighbour classifier (Knn-classifier) determines the class of q by estimating the density around q_v through the K nearest neighbours (Duda and Hart 1973). The latter statement presumes that the feature space is metric, *i.e.* there exists a means of quantitatively determining the distance between two points in the feature space. After determining the K nearest instances and calculating their distance from q, the class is determined by a voting mechanism, namely, *majority voting* or *inverse distance-weighted voting*.

In *Majority voting*, all votes are equal, thus the class with the highest number of instances appearing in the K set of features is voted higher and the q instance is assigned to that class. In *inverse distance-weighted voting* closer neighbours (those with smaller distance) get higher votes, for example, a simple method of calculating the weight of the vote is:

$$vote = \begin{cases} \infty, & \text{if } distance = 0 \\ \dfrac{1}{distance}, & \text{if } distance \neq 0 \end{cases} \qquad \textbf{[eq 3-20]}$$

In order to establish similarity between the neighbours any distance function such as those discussed in Section 2.8 can be used with the respective merits and shortcomings of each choice. Here the L2-distance is used as the similarity measure due to the simple implementation and the wide adoption by the research literature. The L2-distance is given by:

$$d(\vec{q}_v, \vec{t}_v) = \sum_{v=1}^{n} (\vec{q}_v - \vec{t}_v)^2 \qquad \textbf{[eq 3-21]}$$

where, $\vec{t}_v$ is the feature vector of a test image from the data set and $\vec{q}_v$ is the feature vector of a query image.

Neural Networks (NN) are very popular as pattern recognition and machine learning tools in image processing. A comprehensive treat on the subject can be found in (Bishop 2005). As a general outline, neural networks consist of distinctive units called neurons arranged in layers. Each unit applies a linear or non-linear function to an input signal and propagates the output on to the next layer. Individual weightings are applied to the signals as they propagate from one unit to the next one. The learning phase of an NN involves the fine tuning of those weights in order to adapt to the specific intended application. In the context of image processing NNs are very popular as classifiers (Ros, Laurent et al. 2006; Colak and Qahwaji 2007; Ou and Murphey 2007).

Another popular technique is *Adaboost* which is designed to improve the performance of weak classifiers (Freund and Schapire 1997). In Adaboost a weighting factor is assigned to each training sample and a chosen classifier is executed iteratively. During each execution cycle the distribution of weights is updated so that misclassified samples receive an increased weight in the subsequent cycle. Finally, the classifiers are combined using a weighted average and the weighting factors allow the more accurate classifiers to influence more the final decision function. Applications of Adaboost in image classification include (Schwenk and Bengio 1997; Viola and Jones 2001; Wei, Yang et al. 2005)

Support Vector Machines (SVMs) is another family of statistical learning tools used in classification problems. Developed by (Vapnik 1995) SVM method is based on the Structural Risk Minimization principle, which has been proved very efficient in pattern

recognition applications (Burges 1998). SVM classifies an input vector into one of two classes. Classification is performed by separating the vector data using the optimal hyperplane. In a simple case where the sample data are linearly separable, a hyperplane can be visualized as a borderline separating the sample vectors according to their class. Optimality of a hyperplane is defined in terms of maximizing the distance between the closest vector of each class and the hyperplane. For more information the reader is referred to (Chapelle, Haffner et al. 1999). Examples of using SVM for image classification applications can be found in (Wei, Yang et al. 2005; Qahwaji and Colak 2007; Selvan and Ramakrishnan 2007)

Until this point, only the design of the classifier has been discussed. In order to verify this design, it is necessary to quantify a measure for the performance of the classifier. The classifier should be able to accurately classify any possible pattern for which it is designed. The so called *generalization ability* of the classifier (the ability to recognize previously unseen test samples) can be verified by fetching samples that are not included in the design data set.

A popular classifier validation technique is the *leave-k-out* method is used (Devijver and Kittler 1982) (Gong 1986). From a data set of N samples, k samples are used as a test set, whereas the remaining N-k samples are used for training. Then, another set of k samples are used iteratively until all available samples are used once for cross validation. The results are averaged across the n test cases to estimate the prediction performance of the classifier. For the special case of k=1 (unsurprisingly called the leave-one-out method) each sample (image feature vector in this case) is removed from the database and the remaining samples are used for training of the classifier.

A similar approach to the leave-one-out is *jackknifing* which was introduced in (Miller 1964). With jackknifing a random sample is selected for testing and the remaining samples are used for training. The main difference between jackknifing and cross—validation methods like leave-k-out is that the latter is used to estimate the generalization error while the former is used to estimate the bias of the statistic. In simple terms, the statistical bias can be thought as the tendency of the classifier to consistently distort the statistic regardless of the type or the sizes of the training and

test samples. For more information on the jackknife method and application examples the reader is referred to (Kevin L. Priddy and Keller 2005; O'Toole, Abdi et al. 2007; Thomson 2007).

Another popular technique is the *bootstrap* resampling. Instead of iteratively analyzing subsets of the original data sample, the boostrapping method generates a large number of random subsamples from the original data sample with replacement. In order to generate a training data set bootstrapping randomly selects samples from the original data sample set. "Replacement" refers to that, during the selection process, the samples are put back into the original data sample set, thereby allowing for duplicates to appear in the bootstrap training data set. Two major differences between bootstrapping and jackknifing or leave-k-out are: a) the size of the training data set is equal to the size of the original data set (whereas in both jackknifing and leave-k-out is always smaller) and b) the sample replacement allows for a virtual "unlimited" source of data thereby emulating larger data sets. For more information on bootstrap resampling and examples of applications the reader is referred to (Efron and Gong 1983; Gong 1986; Feng, Shi et al. 2004; Kevin L. Priddy and Keller 2005)

The classifier and the associated classification techniques that are used in this thesis have been chosen based on popularity in the texture classification research literature and ease of implementation. As it is discussed in Chapter 6 it would of great interest to investigate the response of the proposed algorithms using different classification schemes. However due to time limitations this is left as a potential direction for future extension of this work. For the classification of a test sample the *Knn*-classifier with majority voting along with the leave-one-out method for cross-validation is used throughout this work. Indicative examples in the research literature include (Ojala, Pietikainen et al. 1996; Van de Wouwer, Scheunders et al. 1999a; Sameer and Sharma 2001; Wilson and Bayoumi 2003; Jafari-Khouzani and Soltanian-Zadeh 2005a; Hiremath and Shivashankar 2008). As discussed in (Van de Wouwer, Scheunders et al. 1999a) and in (Fukunaga 1990) the number of neighbours considered by the kNN-classifier is usually established empirically depending on the application. Here the three nearest neighbours are used (k=3) which corresponds to the scheme used by (Jafari-Khouzani and Soltanian-Zadeh 2005a; Hiremath and

Shivashankar 2008). In all the classification related experiments the results are expressed in terms of the *% classification success* of a system. This is defined as:

$$\%\ classification\ success = \frac{\sum_{j}^{n} classSucce\quad ssRate_{j}}{n}, j = 1,2...,n$$

where, $classSuccessRate_j$ is the percentage of successfully classified samples for the j^{th} texture class.

3.5.3 Benchmark Systems

3.5.3.1 SME-based Energy Signature (SES)

Subband Importance or Subband Energy Signature is one of the most widely applied texture descriptors (see Section 2.7 for examples), primarily due to its proven effectiveness and easy implementation. In (Liang and Kuo 1999) the concept of calculating the subband energy signature based on Significance Map Encoding principles was introduced (see Section 3.3 and (Shapiro 1993) for more information on SME). The paradigm of an SME-based Energy Signature was also adopted by more recent studies such as (Wilson and Bayoumi 2003; Au, Law *et al.* 2007). Here SME-based Energy Signature (SES) is used as a benchmark not only because of its broad adoption by researchers but also because SME is a fundamental component for three of the contributions. Here follows a concise description of the implemented test-bed.

For a predetermined threshold value T the significance N_i of the i^{th} subband is measured according to the number of significant coefficients, as follows:

$$N_i(T) = \left|\left\{c_j \mid c_j \in S_i, c_j \geq T\right\}\right| \qquad \textbf{[eq 3-22]}$$

where S_i denotes the i^{th} subband and c_j the magnitude of the j^{th} coefficient. The importance of the subband is normalised according to:

$$N_{norm,i}(T) = \frac{N_i(T)}{\sum_i N_i(T)}, i = 1,2,...,n \qquad \textbf{[eq 3-23]}$$

The texture vector TV is then composed by the normalised values of subband importance of the subbands:

$$TV = \left\{ N_{norm,1}(T), N_{norm,2}(T) \ldots N_{norm,n}(T) \right\} \qquad \textbf{[eq 3-24]}$$

In order to ensure a fair comparison between the tested systems, the same threshold value has been used for the creation of the Significance Map in SES, localization grid and the Quad-tree operator. Accordingly the same level of decomposition has been maintained throughout the implementations for uniformity.

In order to compare two images Q and I with respect to their texture vectors TV_Q and TV_I respectively, SES uses the L_1-distance of their texture vectors:

$$d_{L_1}(Q,I) = \left(\sum_{i=1}^{n} w_i \cdot \left| TV_Q - TV_I \right| \right) \qquad \textbf{[eq 3-25]}$$

where w_i is the weighting factor for the i^{th} subband. The value for w_i is calculated as follows:

$$w_i = \frac{1}{nc_i} \qquad \textbf{[eq 3-26]}$$

where nc_i is the number of DWT coefficients in subband i.

3.5.3.2 LBP

A second benchmark was implemented based on the Local Binary Pattern $LBP_{P,R}$. Although this algorithm operates on pixel domain is and thus not directly comparable to the algorithms discussed in this work it remains one of the state-of-the-art propositions. Furthermore the LBP algorithm is based on the concept of an autoregressive occurrence model which is also central to the LWCP algorithm presented here. Here follows a description of the LBP test-bed implementation.

$LBP_{P,R}$ provides for an operator pattern of any radius (defined by subscript R) and number of sample bits (defined by subscript P). A set of 8 coefficients with a distance of 1 cell are used for the construction of the LWCP operator. Therefore, in order to construct a comparable test bed the $LBP_{8,1}$ version is used as the closest in terms of complexity and computational cost. This corresponds to the original version of LBP as described in (Ojala, Pietikainen *et al.* 1996). In the LBP texture operator a 3×3 pixel cluster is thresholded at the value of the central pixel (see Figure 3.16). Let c_0 to c_7 be the grey scale values of the eight peripheral pixels of the cluster and

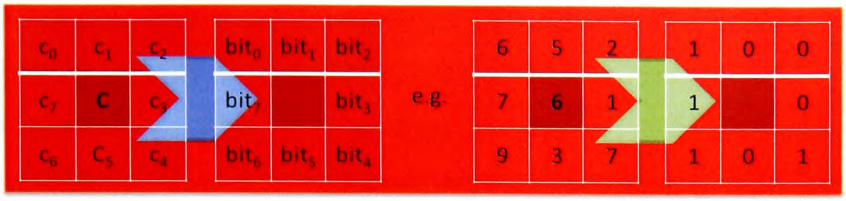


Figure 3.16: Graphical representation of the construction of the LBP operator

C the grey scale value of the central pixel. An 8 bit pattern is constructed by assigning each surrounding pixel the value of 0 or 1 according to the following rule:

$$bit_i = \begin{cases} 0, \text{if } c_i < C \\ 1, \text{if } c_i \geq C \end{cases}, \forall i = 0,1,2...7 \qquad \textbf{[eq 3-27]}$$

The LBP operator is then obtained by assigning each bit to a binomial weight (see fig 3.16), as follows:

$$LBP = bit_7 \cdot 2^7 + bit_6 \cdot 2^6 + bit_5 \cdot 2^5 + bit_4 \cdot 2^4 + bit_3 \cdot 2^3 + bit_2 \cdot 2^2 + bit_1 \cdot 2 + bit_0 \qquad \textbf{[eq 3-28]}$$

3.5.4 Wavelet Filter-banks

Wavelet Transformation requires two parameters to be defined for the experiments: the maximum level of decomposition and, the wavelet filter bank. The level of decomposition has been defined separately for each algorithm and is discussed in the respective sections. The filter-banks that have been used for the experiments are the Daubechies (5,3) and the Daubechies (9,7) (see Tables 3.2 & 3.3 for the respective filter coefficients) (Antonini, Barlaud *et al.* 1992; Daubechies 1992). Different opinions exist in the research literature (Chang and Kuo 1993; Unser 1995) regarding the effect of different filter-banks on wavelet-based texture extraction and studies have shown that certain characteristics of filter-banks do affect texture (Mojsilovic, Popovic *et al.* 2000a). In this work, the two filter-banks are considered for the following reasons:

- They have been widely used in the research literature for wavelet-based image retrieval experiments therefore offer more comparable results.
- The particular filter-banks are proposed by the JPEG2000 international standard for loss-less (5,3) and lossy (9,7) image compression (Rabbani and

Joshi 2002a). This helps towards the original aim of this work to introduce techniques that can be applied on compressed images.

- In (Mojsilovic, Popovic *et al.* 2000a) these filters are successfully tested for their suitability in texture characterization applications.
- The two filter-banks performed in a very similar manner therefore only the experimental results obtained using Daubechies (9,7) are presented in the empirical analysis sections for brevity.

Table 3.2: Analysis $h_0(n)$ and Synthesis $g_0(n)$ filter coefficients for the Daubechies (9,7) filter bank.

Synthesis		Analysis	
n	Low-pass, $h_0(n)$	n	Low-pass, $g_0(n)$
0	+ 0.6029490182363579	*0*	+ 1.1150870524569940
±1	+ 0.2668641184428723	*±1*	+ 0.5912717631142470
±2	- 0.0782232665289879	*±2*	- 0.0575435262284996
±3	- 0.0168641184428749	*±3*	- 0.0912717631142495
±4	+ 0.0267487410809760		

n	High-pass, $h_1(n)$	n	High-pass, $g_1(n)$
-1	+ 1.1150870524569940	1	+ 0.6029490182363579
-2,0	+ 0.5912717631142470	0,2	+ 0.2668641184428723
-3,1	- 0.0575435262284996	-1,3	- 0.0782232665289879
-4,2	- 0.0912717631142495	-2,4	- 0.0168641184428750
		-3,5	+ 0.0267487410809760

Table 3.3: Analysis $hX(n)$ and Synthesis $gX(n)$ filter coefficients for the Daubechies integer (5,3) filter bank.

N	$h_0(n)$	$g_0(n)$	n	$h_1(n)$	n	$g_1(n)$
0	+3/4	+1	-1	+1	1	+3/4
±1	+1/4	+1/2	-2,0	-1/2	0,2	-1/4
±2	-1/8				-1,3	-1/8

3.6 Empirical analysis – Algorithm response with incomplete set of DWT data

The processing required for both retrieval and classification is proportional to the actual volume of data. It is therefore desirable to be able of extracting all necessary feature information from a smaller data set without compromising accuracy.

Furthermore, considering the compression and feature extraction problem jointly, the algorithms may have to cope with images that are already compressed using a DWT decomposition tree with a specific layer depth. Another option would be to further process the DWT tree in order to obtain more layers of decomposition, provided that this is allowed by the image resolution. However, this would "move" the feature extraction algorithm to the left of point C in figure 1.2, thereby disqualifying it from operating in "compressed domain". To this end, in this first part of the empirical analysis, the effect that different levels of available wavelet information have on the accuracy of the algorithms is investigated.

All methods use the DWT decomposition tree as the source data. In most cases, the decomposition tree is further processed in order to obtain the Significance Map of each subband. To this end, two approaches for reduction of the processed data are investigated in this section.

The first approach is to exclude one or more layers of decomposition from the calculation process. Based on the particular characteristics of each method, the layers are removed starting either from the coarse side or the detail side of the decomposition tree. This also provides an indication of the extent to which coarse image features and finer detail image features contribute to the discrimination ability of each method.

The second approach is applied after the calculation of the Significance Map. After establishing the importance of each subband, the least important subbands are excluded. Since it is unfeasible to pre-determine which layers contain the least important subbands, this approach can only be applied to methods that perform per-subband and not per-layer calculations. This analysis provides an indication of the extent to which subband importance contributes to the discrimination ability of each method.

3.6.1 Localization Grid – Effect of using different grid sizes & excluding coarse DWT layers (Experiment 1)

In the Localization Grid algorithm, the size, hence the granularity of the grid, affects the volume of data to be processed. A grid with high granularity, results in higher processing requirements than a grid with lower granularity. As it was established in Section 3.4.1.1, the size of the localization grid also affects the accuracy of the algorithm. It is therefore desirable to determine the trade off between accuracy and speed for different localization grid sizes.

In order to determine the effect of different localization grid sizes, the first experiment was performed on images that were decomposed to the maximum possible level of DWT, and all the subbands were included in the signature. The grid sizes that were used for experimentation are those determined in Section 3.4.1.1. For database B, where the size of images is 512×512, the dimensions of the largest subband are 256×256. As it was discussed in Section 3.4.1.1, the smallest applicable subband size is 16×16. Therefore, the images can be decomposed up to the 5^{th} level (subband size 16×16).

Due to their larger size, finer subbands can accommodate a larger range of localization grid sizes than coarser subbands. If a dyadic approach is used when deciding on different sizes of localization grids, dimensions are increased only by doubling. The possible Localization Grid sizes that correspond to the available subband sizes are shown in table 3.4. The chosen subband sizes correspond to each of the first four levels of decomposition.

Table 3.4: Possible Localization Grid sizes for given subband sizes

Subband size		Localization Grid size		Cell size		
X	Y	min	max	max	min	Size Permutations
16	16	4×4	8×8	8×8	4×4	2
32	32	4×4	16×16	16×16	4×4	3
64	64	4×4	32×32	32×32	4×4	4
128	128	4×4	64×64	64×64	4×4	5
256	256	4×4	128×128	128×128	4×4	6

Table 3.5: Experimental configurations for investigation of different localization grid sizes and number of decomposition layers used

Layers	min subband size	Localization Grid Sizes tested
5	16×16	4x4 8x8
4	32×32	4x4 8x8 16x16
3	64×64	4x4 8x8 16x16 32x32
2	128×128	4x4 8x8 16x16 32x32 64x64
1	256×256	4x4 8x8 16x16 32x32 64x64 128x128

It is also desirable to investigate the effect of using fewer layers of decomposition, in order to achieve savings in processing cost. Therefore, the experimental configurations shown in table 3.5 are used. The localization grids were chosen so that the guidelines described in Section 3.4.1.1 are satisfied.

The classifier results shown in Figure 3.17 indicate that performance deteriorates for large localization grid dimensions. This can be attributed to the fact that the higher granularity makes the algorithm very sensitive to local subband details. As a result minor spatial variations between similar images generate different Localization Tree representations. The trend of higher performance for smaller localization grid dimensions reverses at the size 4×4, which is also the smallest. Grid size is inversely proportional to cell size. In the case of a small grid the cell size is therefore large. From this it can be theorised that when the selection criteria for an active cell are not met a lot of information is discarded, thus resulting in performance drop. Adding information from more layers of decomposition strengthens the discrimination ability of the algorithm with each additional layer contributing between 3% and 10%.

This is attributed to the additional multi-scale detail information that becomes available to the descriptor. The highest accuracy is observed for Localization Grid dimensions of 8×8 and 16×16 at 85.7% and 83.9% respectively. Although the 8×8 and the 16×16 localization grids have similar performance, the 8×8 version is preferable due to lower processing cost and more compact representation.

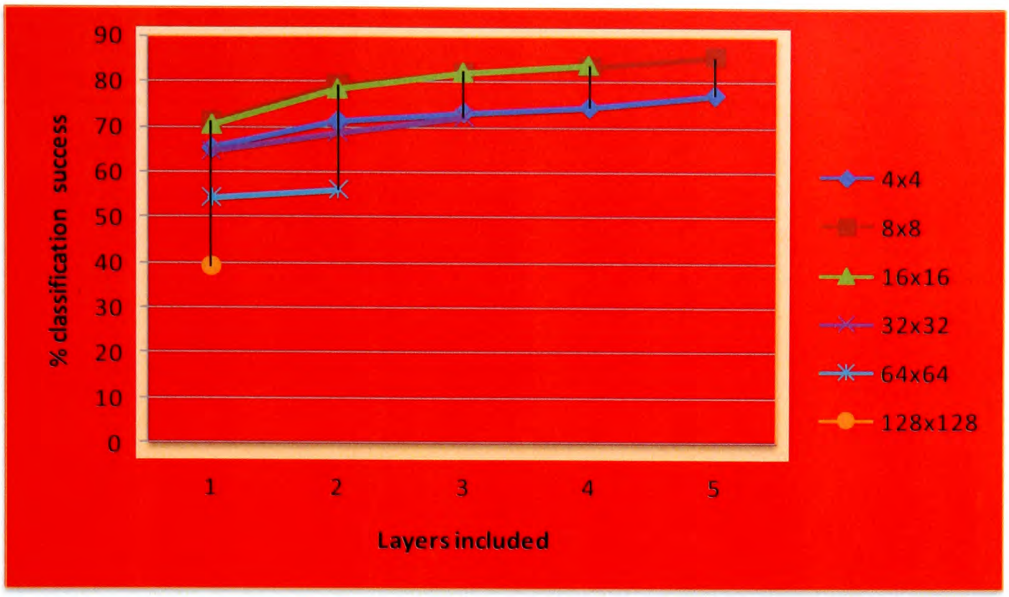


Figure 3.17: Localization Grid; effect of variable grid sizes and number of decomposition layers used (starting from level 0).

3.6.2 Localization Grid – Excluding detail DWT layers (Experiment 2)

In experiment 2 the effect of using fewer high frequency layers is examined. As it was established in experiment 1, localization grid sizes of 4×4 and 8×8 cells can be accommodated by all layers of decomposition. Therefore, the experiment is performed using those two sizes. Based on the inference that, the concentration of significant coefficients is higher in higher levels of decomposition (towards LL subband), the process starts from using only layer 5 information (coarser) and gradually adding lower (finer) layers of decomposition to the creation of the signature. As it is shown in Figure 3.18 the accuracy of the algorithm increases as more detail layers are added. Furthermore, the rate of improvement is higher between the first three layers (detailed) and it is lower between the last two (coarse). Observation of Figure 3.17 and 3.18 also shows that the accuracy achieved for a single high frequency layer (Figure 3.17) is almost double than for a single low frequency layer (Figure 3.18) for the same localization grid sizes. This leads to the conclusion that high frequency layers contain more useful information for the descriptor than lower frequency layers. In other words inclusion of the finer detail image information which is conveyed by the higher frequency subbands improves the

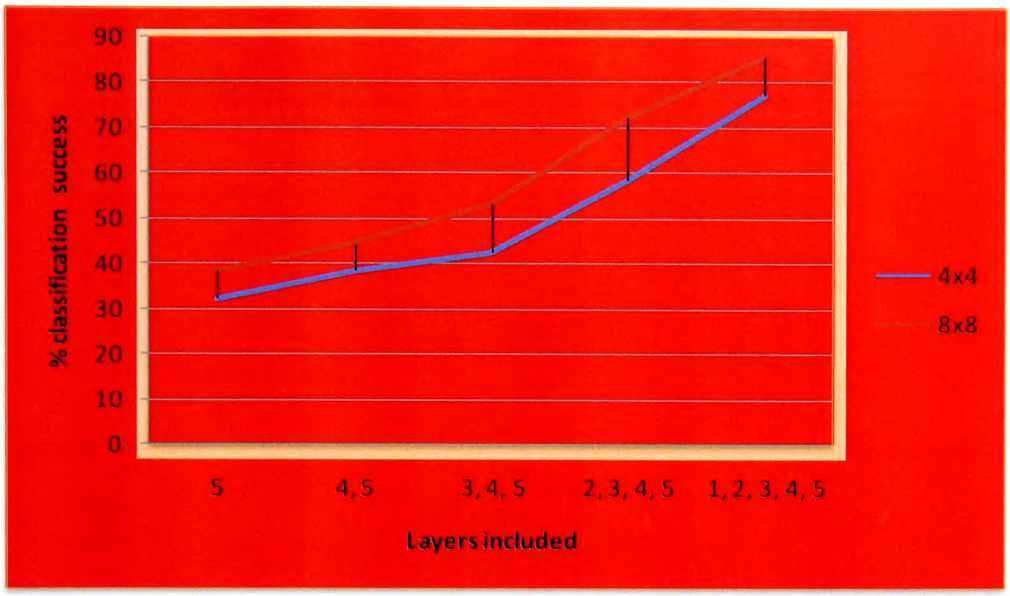


Figure 3.18: Localization Grid; effect of variable number of decomposition layers used (starting from level 5).

accuracy of the descriptor at a higher rate than the coarse image information conveyed by the lower frequency subbands. In physical terms this reflects the fact that the differences between texture patterns become more visible and distinguishable as the resolution of an image becomes larger, thereby exposing more information about finer details.

3.6.3 Localization Grid – Select subbands based on their importance (Experiment 3)

A different approach was used in the third experiment, in order to investigate the possibility of increasing the discrimination ability of the algorithm. In this experiment, the Significance Map of the full decomposition tree is used in order to determine the most *important* subbands (those with the highest number of significance coefficients).

After creating the subband importance map, the localization grids of the most important subbands are used for the creation of the texture feature signature. The experiment starts using only the most important subband. The less important subbands are included gradually until all available subbands are used. An 8x8 localization grid size is used, since it was found to be the best performing in the

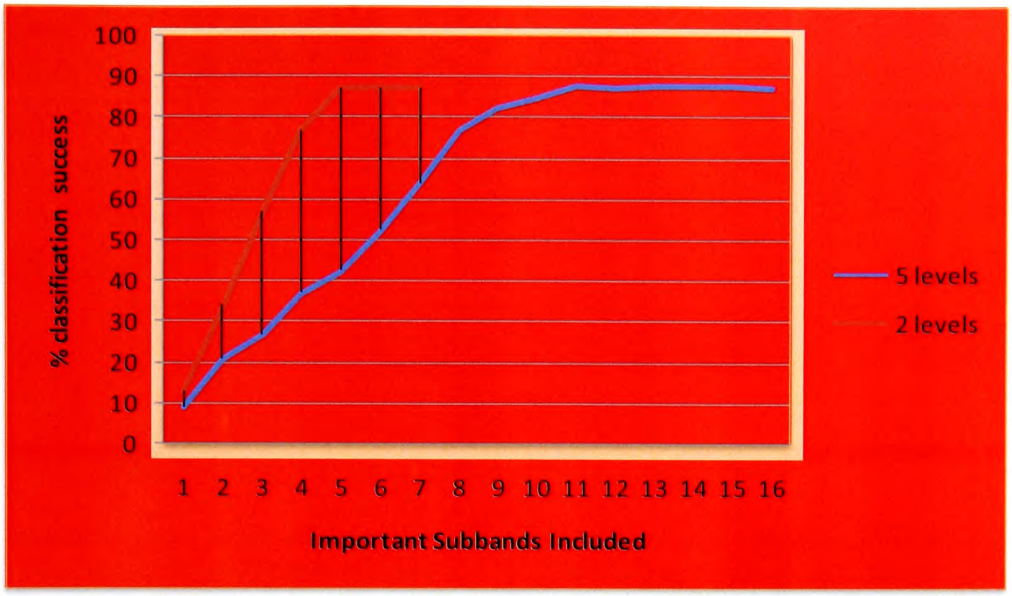


Figure 3.19: Localization Grid; effect of variable number of important subbands used (starting from most important). Results for 2- and 5-level decompositions.

previous experiments. According to the results which are shown in Figure 3.19 the maximum accuracy for 5 decomposition layers is achieved with 11 subbands (87.8%) and for 2 decomposition layers it is achieved with 5 subbands (87.4%). Therefore, in both cases the accuracy saturates when approximately 70% of the available subbands are included.

3.6.4 Quad-Tree – Excluding detail DWT layers (Experiment 4)

Starting by processing data from one layer only, this experiment investigates the effect of gradually increasing the Quad-tree size by adding more layers of decomposition, hence Quad-tree nodes. In order to have available five levels of decomposition, the super set of the database described in Section 3.6.1 is used. The LL (layer 5) subband contains 16×16 coefficients.

The average classifier success results shown in Figure 3.20 indicate that using only the LL layer does not provide sufficient accuracy which is expected since this operation is essentially a one-to-one comparison of two Significance Maps. On the opposite side, the accuracy drops steeply (>10% per layer) when more than four layers are added as the tree becomes larger, more complex and thus prone to noise. Peak accuracy is achieved with three layers (65,1%).

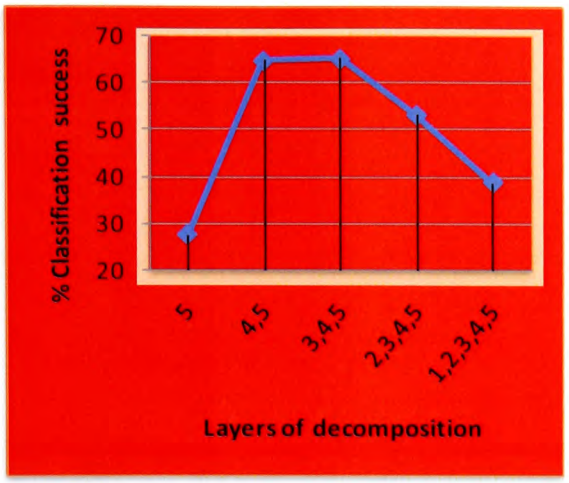


Figure 3.20: Quad-tree; effect of variable number of decomposition layers used (starting from level 5).

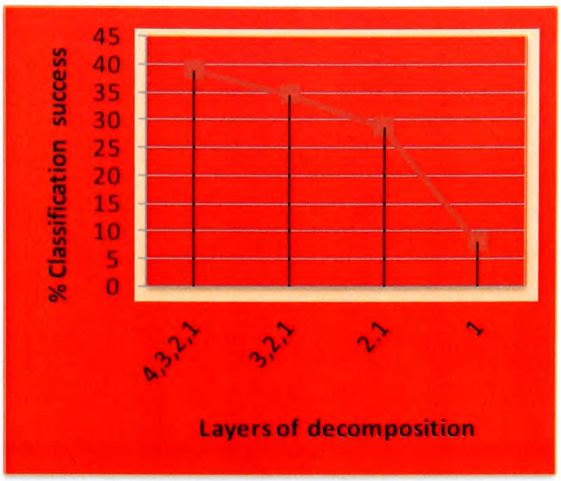


Figure 3.21: Quad-tree; effect of different root node locations.

3.6.5 Quad-Tree – Excluding coarse DWT layers (Experiment 5)

Until this point it has been implied that the root nodes of the quad-tree are located at the LL subband of the maximum layer of decomposition. Relocating the root nodes to lower layers of decomposition is studied in this experiment. The methodology of experiment 1 for the exclusion of coarse DWT layers is used.

The classifier results are shown in Figure 3.21. Using a single layer of decomposition makes the descriptor almost identical to the Significance Map hence very selective and this is reflected by the very low performance which was recorded. In a similar way as in the previous experiment, the accuracy drops steeply when more than four layers are utilised. The best results are obtained for 3-4 layers of decomposition.

Concluding the algorithm shows high sensitivity in the number of decomposition layers that are used which is expected due to the drastic effect of each layer to the size and constitution of the Quad-tree operator.

3.6.6 LWCP – Excluding coarse DWT layers (Experiment 6)

The LWCP algorithm analyses the detail content of the DWT tree in order to extract texture patterns. Therefore, experiment 6 starts with the inclusion of finer detail layers. Coarser detail layers are added gradually in order to detect the saturation

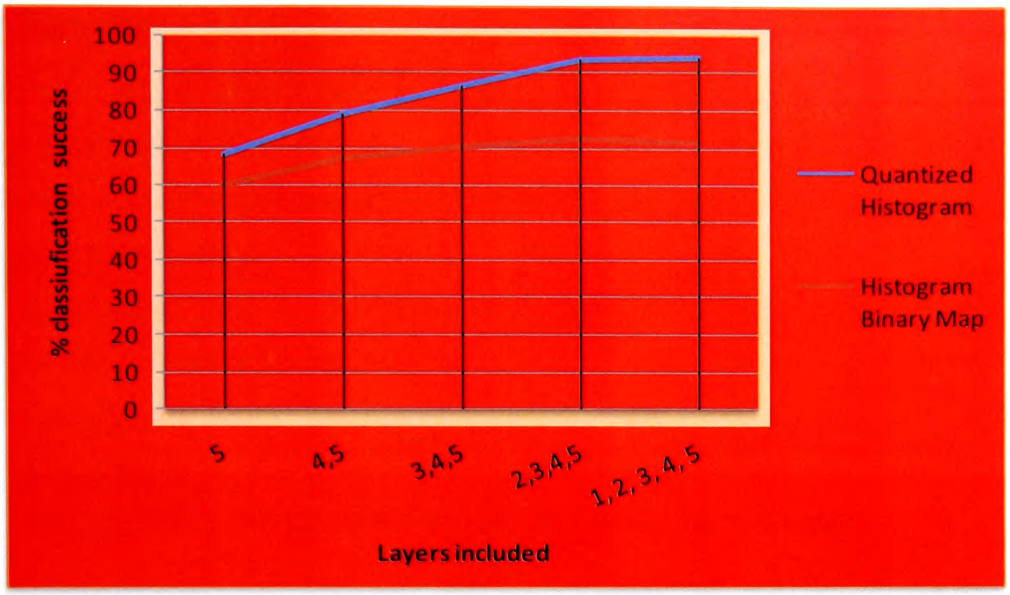


Figure 3.22: LWCP; Effect of variable number of decomposition layers used (starting from level 5).

point, *i.e.* the point where the accuracy of the algorithm no longer improves, or even deteriorates. The experiments are repeated for each of the proposed histogram creation techniques. For the *Quantized Histogram* signature 64 levels of quantization are used for each bin. For the *Histogram Binary Map* the threshold is set so that at least 30% of the bins qualify.

The average classifier results are shown in Figure 3.22. The performance of both signature construction methods increases, as more layers are added into the calculation of the feature. The maximum accuracy is achieved by Quantized Histogram when all layers are included (94%). With four layers the accuracy of the algorithm is marginally lower (93.7%). Maximum accuracy for Histogram Binary Map is with four layers and a small decrease in accuracy with five layers of decomposition. The detail subbands appear to possess significant texture information since even with a single decomposition layer the accuracy of the algorithm is about 70% (quantized histogram). The Histogram Binary Map gives on average 15% lower accuracy than Quantized Histogram since a considerable amount of data are discarded resulting in a signature which is only $1/7^{th}$ in size.

3.7 Empirical analysis – Classification & retrieval

The second set of experiments evaluates the performance of the algorithms in texture classification (Experiments 7 and 8) and generic image retrieval (Experiment 9). Building upon the conclusions of the previous sections, the following versions of the algorithms are used:

- Localization Grid: an 8x8 grid is used and 70% of the subbands are selected based on their importance.
- Quad-trees: Three layers of decomposition are used. All root nodes lie in the LL subband.
- LWCP: Quantized Histogram is used for the signature creation.

3.7.1 Classification – Texture perceptual characteristics (Experiment 7)

Having established the effect of different parameters on the algorithms, the classification ability of the best performing versions is now evaluated against different perceptual characteristics of textures. The experiments are designed to highlight the sensitivity of the algorithms to different perceptual characteristics of textures as defined in Section 3.5.1.2. Results from the two test-bed systems are also provided for comparison (see Section 3.5.3). Database C is segmented into four sub-groups with similar perceptual characteristics. The classifier results are shown in table 3.6.

Table 3.6: Classification Performance; Effect of different texture perceptual characteristics. Results expressed in % classification success.

Method / Group	Localization Grid		Quad-trees	Quad-tree Patterns	LWCP	LBP	SES
	4×4	8×8					
C.1	77.1	90.0	85.1	94.3	99.0	99.0	75.0
C.2	73.0	85.3	52.2	89.4	95.5	98.2	73.0
C.3	73.1	86.0	44.6	89.0	94.9	95.5	74.1
C.4	72.4	81.7	37.5	79.7	85.2	82.7	70.2

Since all texture descriptors primarily rely on the directional frequency responses (effectively directional edge periodicity information) of DWT it is unsurprising that all

algorithms perform best with strongly consistent patterns. The Quad-trees algorithm is very selective in terms of texture perceptual characteristics, performing well only with strongly consistent patterns. This validates the earlier observation that the algorithm is sensitive

SES and Localization Grid are the least affected from variations on the perceptual properties of the textures, which can be attributed to the fact that both algorithms are based on subband-wide statistics rather than coefficient-level local analysis of the other techniques. This however comes at a cost since the overall performance of the two algorithms is weaker. Overall the results indicate that the LWCP and the LBP algorithms are the best performing. LWCP is the best performing algorithm for varied patterns and the LBP algorithm is marginally better in detecting random and quasi consistent structured patterns.

3.7.2 Classification - Complete texture database (Experiment 8)

The texture classification performance of the algorithms is evaluated using Database B. The normalized classifier results are shown in table 3.7.

Table 3.7: Classification Performance. Results expressed in % classification success.

Method	Localization Grid		Quad-trees	QT Patterns	LWCP	LBP	SES
	4×4	8×8					
Accuracy	75.7	87.3	65.2	87.8	94.0	94.7	75.0

LWCP and LBP are overall the best performing algorithms for texture classification consistently with experiment 7. Similarly, Quad-trees algorithm has inferior accuracy than the remaining algorithms. The more compact form of Quad-tree Patterns has considerably higher performance than Quad-trees indicating that the compact descriptor which focuses on local multi-scale information provides a much better representation of texture. Although Localization Grid gives inferior results to other techniques, the drop in performance could be justified in applications where the considerably simpler implementation and lower processing requirements for the construction of the feature signature give good reason for lower accuracy.

3.7.3 Generic Retrieval (Experiment 9)

The last experiment evaluates the algorithms on generic (natural) image retrieval using the methodology described in Section 3.5.2.1 on database A. Results from the two test-bed algorithms are also presented here for comparison. The term generic retrieval is used here to indicate that the database has a very diverse content (broad image domain). The results are shown in Table 3.8 and include the variance as an indication of how the accuracy fluctuated between different queries.

Table 3.8: Image Retrieval accuracy

Query		Average Precision (%)	Variance
% correct matches	**Loc. Grid**	75.1	1.1
	Quad-trees	70.0	4.0
	Quad-tree Patterns	61.2	0.9
	LWCP	62.3	0.8
	SES	55.2	1.1
	LBP	65.6	1.5

Localization Grid and Quad-trees have the highest performance highlighting the fact that both algorithms incorporate some information about regional distribution and shape which seems to suit the very diverse content of database A. The high variance of quad-trees indicates again the sensitivity of the algorithm on diverse content. On the other end, LWCP and Quad-tree patterns showed lower precision but still directly comparable to the benchmarks accuracy.

3.8 Conclusions

Building upon the identified shortcomings of existing research, this chapter introduced four methods for texture characterization in wavelet compressed domain and, their application in texture classification and image retrieval. The issue addressed is primarily accuracy although all techniques are using compression-compatible wavelet parameters and are based on, or can be extended to, the SME paradigm which allows them to be applied directly in compressed domain. Part of the work presented in this Chapter on the Localization Grid and Quad-trees algorithms

has been presented in national and international refereed conferences (Voulgaris and Jiang 2001c; Voulgaris and Jiang 2001d; Voulgaris and Jiang 2002c; Voulgaris and Jiang 2002a).

The first approach, called the Localization Grid, is a low-processing cost enhancement of existing Subband Energy Signature (SES)-based techniques which adds locality information to the texture descriptor. After calculation of the Significance Map of a DWT of an image, Localization Grid divides each subband into regions using a pre-defined grid. The relevant distribution of significant coefficients determines which regions are going to be used for the calculation of the signature. A grid is constructed for the query image and is used afterwards as a template for the calculation of test image signatures.

The performance improvement is between 10% and 20% over traditional SES for both classification and retrieval. Experiments also indicate that processing savings can be achieved by using 70% of the important subbands for the calculation of the descriptor without reducing the accuracy. This is particularly helpful in compressed databases where importance can be determined at the decoding stage.

Two system parameters were identified, which could potentially affect accuracy: the size and the granularity of the localization grid. Grid sizes of 1/64 to 1/32 of the image size were proven to be the best performing. However, it must be noted that the effective size and granularity of the localization grid could potentially be affected by changing the scale of the image. On the other hand, the plurality of textures that were used for testing ensure that the conclusion accommodates equally smooth, rough and mixed textures.

The following two techniques, called Quad-tree and Quad-tree Patterns, investigated the use of quad-trees directly as a feature for texture-based retrieval.

Quad-tree signature is constructed by designating the most significant coefficients of the LL subband as root nodes and building a quad-tree for each one of them according to the significance of the leaf nodes. The set of generated quad-trees forms the feature signature. Comparison between images is performed by XOR-ing

their signatures. The quad-tree is dependent on local and global significance characteristics of images, thus conveying information about shape and texture. The experimental results showed that Quad-trees, the first of the two techniques, has very selective behaviour with regards to the perceptual characteristics of the textures. This issue was attributed to the leaf nodes of the multi-layer tree structure which get saturated with high-frequency information. The latter is thus manifested to the feature signature as noise.

The quad-tree algorithm and the paradigm of the embedded zero-tree wavelet coding (Shapiro 1993) form the basis for the construction of the Quad-tree Patterns operator. The information obtained by two consecutive levels of decomposition, one root node and its four immediate descendants, are encoded using a tri-state symbolization. Quad-tree Patterns aims towards resolving the aforementioned issue by extracting two-layer quad-tree features. The algorithm gave much better and consistent results for both classification and retrieval.

The last technique, called (Local Wavelet Coefficient Patterns) LWCP, is a novel approach extracting texture from wavelets via an autoregressive occurrence model. The model uses a 3×3 binary mask in order to detect micro patterns within subbands. Subband histograms are generated in order to form the feature signature. The method showed very good performance compared to both the benchmarks and the remaining algorithms. An alternative version of the algorithm with a smaller signature size was tested (32 bytes/subband over 224 bytes/subband) for savings in storage and processing during similarity testing. The smaller version was on average 15% less accurate for approximately $1/7^{th}$ of the size.

Since Quad-tree Patterns and LWCP exploit the micro-features and the pattern structures contained in the DWT subbands, they were more suitable candidates for texture classification than for generic image retrieval. Localization Grid and Quad-trees algorithms take into account a combination of global and local characteristics of an image, thereby incorporating knowledge on how information is distributed within the image. Consequently, they were more suitable for generic image retrieval applications. This also supports the assertion that multi-feature and multi-metric systems are more suitable for generic CBIR systems.

ALGORITHMS RESPONSE WITH INCOMPLETE SET OF DWT DATA

For the Localication Grid algorithm, two cases were studied, either to omit some decomposition layers or to omit subbands based on their importance. Although the concentration of significant coefficients is lower in detail subbands (lower decomposition layers), experimental results indicated that, omitting subband triplets (*HL, LH, HH*) which correspond to any decomposition layer, led to significant accuracy deterioration, when using the Localization Grid technique.

On the other hand, if the filtering is performed based on per-subband importance, significant savings in processing can be achieved. It is noteworthy that accuracy saturates when approximately 70% of the most important subbands are included in the calculation of the localization grid. By taking into account that, calculating the importance of a subband is much faster than calculating the localization grid signature, it is apparent that the gain in processing time can be considerable.

With quad-trees, the effect of limiting the quad-tree levels was investigated either, by relocation of the root nodes to higher layers or, by limiting the depth of the generated tree. A quad-tree with a depth of fewer levels than the full DWT was generated by omitting finer detail levels of decomposition.

The outcome of the experiments was that, starting from the highest decomposition level, the accuracy of the quad-tree algorithm increases for up to three-levels of tree depth. It is interesting that, contrary to Localization Grid, adding the higher detail levels of decomposition negatively affects the accuracy. The same applied when the root nodes were relocated to higher levels of decomposition.

In both cases, the algorithm appears to be very selective in local detail variations as finer detail information is added. This can be attributed to the dependence of the accuracy of the Quad-tree technique on the absolute spatial position, hence distribution, of the leaf nodes.

For LCWP, two scenarios were investigated: a) reduce the number of decomposition levels that are processed, b) apply a threshold to the generated histogram

signatures, in order to provide a binary version (Histogram Binary Map) that requires significantly lower storage space and image comparison processing time. Empirical analysis yields that, using either histogram generation technique, only very coarse ($>5^{th}$) layers of decomposition can be omitted, without significant losses in the accuracy of the algorithm.

For Histogram Binary Map, the loss in accuracy over Quantized Histogram reaches 15%; hence it is only justified in situations where accuracy must be traded-off for smaller signature storage space and lower processing requirements.

CLASSIFICATION AND RETRIEVAL

The first set of texture classification empirical analysis examined the sensitivity of the algorithms to different texture perceptual characteristics. Overall, the most promising results are achieved using the LWCP algorithm. Notably, the Localization Grid algorithm is affected the least by variations in the perceptual characteristics of textures, whilst lacking the accuracy of the remaining methods. Out of the two quad-tree-based techniques, only the results obtained using Quad-tree Patterns are comparable to the remaining methods.

For generic image retrieval, Localization Grid and Quad-tree algorithms were the most accurate followed by the LWCP. Quad-tree results show a greater variance, which indicates a higher sensitivity to the contents of the samples. Both Localization Grid and Quad-trees algorithms incorporate global distribution information and hence are more sensitive to the exact spatial distribution of high frequency components within an image. This makes the algorithms less accurate in distinguishing patterns, but more accurate in characterising the edge-derived features of an image as a whole. Notably, the algorithms perform worse in generic image retrieval than in texture classification, which is due to the inherent increased difficulty of the retrieval problem over the more clearly defined ground truth of classification.

Chapter 4: Rotation Invariant Texture Characterization using Wavelets

4.1 Abstract

In this chapter, the issue of narrowing the sensory gap is addressed starting from a model for rotation invariant texture characterization in wavelets domain. The model is used to predict the edge information included in the wavelet coefficients of the detail subbands for several orientations. In order to create rotation invariant texture descriptors, the model is used to design two algorithms employing either a combination of subband statistics or local coefficient information.

4.2 Introduction

The wavelet-based texture feature extraction techniques presented in the research literature can be broadly categorised in two groups; those that are based on the orthogonal wavelet transform and those that are based on variations of the wavelet transform employing, for example, Gabor filters or the ridgelet transform. Representative examples from both categories can be found in the Section 2.7.4.

The considerably lower complexity of orthogonal decomposition makes it more convenient for practical implementations. Additionally, feature extraction algorithms based on the orthogonal wavelet transform are better candidates for manipulation of databases of pre-compressed images, since wavelet-based compression codecs commonly use orthogonal decomposition.

An issue with regards to multi-orientation analysis using orthogonal discrete wavelet transform lies in the output of the low- and high-pass filtering process. The output of the high-pass filtering stage contains information about the horizontal, the vertical and the diagonal edges of the image. Thus the subbands do not provide directly accessible information for angles between 0° and 45°.

Although the subbands of the wavelet decomposition contain edge information about specific orientations, this information is sufficient for the reconstruction of the original image. Therefore, the problem of constructing a rotation invariant descriptor lies, primarily, on the appropriate interpretation and exploitation of the information conveyed by the detail subbands. Since the coarse (low-pass filter) subband does not contain any directional edge information, it can be disregarded.

The most frequently cited work which suggests a resolution to this issue is (Porter 1997). Porter presents a method for texture extraction in wavelets domain using the L_1-norms of opposite (vertical and horizontal) pairs of directional subbands. To achieve rotation invariance, the contribution of each subband inside a pair is regulated by a weight. However, as it was discussed in Chapters 1 and Chapter 2, reportedly the rotation invariant version of the pixel-based LBP algorithm (Ojala, Pietikainen et al. 2002b) outperforms Porter's approach and is still overall considered a state of the art approach.

To this end, this chapter introduces techniques designed to narrow the sensory gap of texture extraction by achieving rotation invariance and compatibility with wavelet-based compression algorithms, at the same time, without compromising accuracy.
To achieve that the techniques are based: a) on orthogonal DWT to ensure compatibility with compression algorithms without compromising accuracy and, b) on a modified autoregressive occurrence model, building upon the paradigm of the LWCP algorithm introduced in the previous chapter.

Building upon the observation by (Li and Jay Kuo 1999) that rotation of the orthogonal wavelet subbands at specific orientations can be performed using only a few operations, a novel technique for mapping wavelet coefficient data on a rotated locus called Wavelet Isometric Model is introduced in Sections 4.3 and 4.4. The model is applied to design two techniques (Section 4.5) for rotation invariant wavelet-based texture extraction, as follows: The first approach is an augmented rotation invariant form of subband statistics, namely the variance and the energy histograms. The second approach combines multi-directional coefficient information for estimation of the edge information at different rotation angles.

4.3 Wavelet Isometric Operators

Due to the orthogonallity of the wavelet subbands rotations of 90, 180 and 270 degrees of a wavelet transformed image can be performed with only a few operations (Li and Jay Kuo 1999). Table 4.1 shows the necessary operations for each angle of rotation.

Table 4.1: Wavelet Isometry Operators

Image Rotation	Required Wavelet Processing	
	Exchange bands LH and HL	Sign change for bands:
90 degree rotation	Yes	HL, HH
180 degree rotation	No	LH, HL
270 degree rotation	Yes	LH, HH

Note that the operators of table 4.1 regard the magnitude of the coefficients only. For a 90 degrees rotation the coefficients of each subband have to be physically rotated by 90 degrees too.

In order to derive the model for a rotation invariant descriptor, let us consider how a single coefficient is affected by the rotation process. Let, HL_x and HL_y be the coordinates of a wavelet coefficient in the HL subband of the first level of decomposition as shown in Figure 4.1. Firstly, after the actual 90° rotation of the coefficients, the coefficients of the HL subband are replaced by those of the LH subband as dictated in Table 4.1. Secondly, the signs of the coefficients of the HL subband are swapped. Therefore, the 90° rotation will result in the new coefficient at location (HL_x, HL_y) being equal in magnitude to the old coefficient at the same location as the LH subband, but with the opposite sign.

In a similar way, the coefficient with coordinates (LH_x, LH_y) will be equal with the coefficient which was located at the same coordinates in the HL subband before the rotation. As it is shown in Table 4.1 there is no need for a sign change in subband LH. Finally, after the rotation, a coefficient in the HH subband will have the same magnitude but opposite sign.

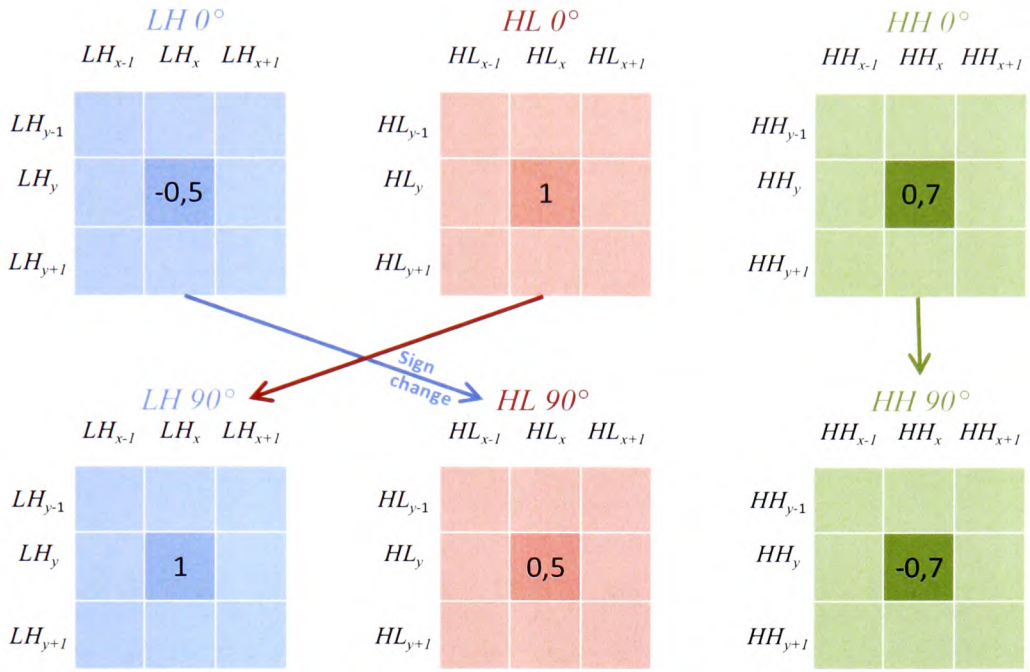


Figure 4.1: Effect of a 90° rotation on a single coefficient of a) the HL subband b) the LH subband, and c) the HH subband.

Figure 4.2 depicts a different perspective of the isometric operations with regards to the sign changes. Here the subbands are viewed as vectors with directions derived from the respective subband, *i.e.* the vertical detail subband is represented by a downwards pointing arrow. As it can be seen there is a relationship between the direction of the vectors after the initial rotation and the change in the sign of the coefficients. As a rule of thumb, the sign of a coefficient changes whenever there is a change in the direction of the respective subband vector.

Let C_{xy}^{HL} be the coefficient at coordinates (x,y) in subband HL and C_{xy}^{LH} the coefficient at the respective location in subband LH. Observation of how C_{xy}^{HL} changes during image rotation leads to the following two conclusions:

a) The magnitude of the coefficient changes periodically between C_{xy}^{HL} and C_{xy}^{LH} with a period of $\pi(180°)$.
b) For each of those two alternating magnitudes the sign of the coefficient changes periodically from + to – every π rads $(180°)$, or alternatively, the signs of the two possible magnitudes change with a period of π rads $(180°)$ and a phase difference of 90° between them.

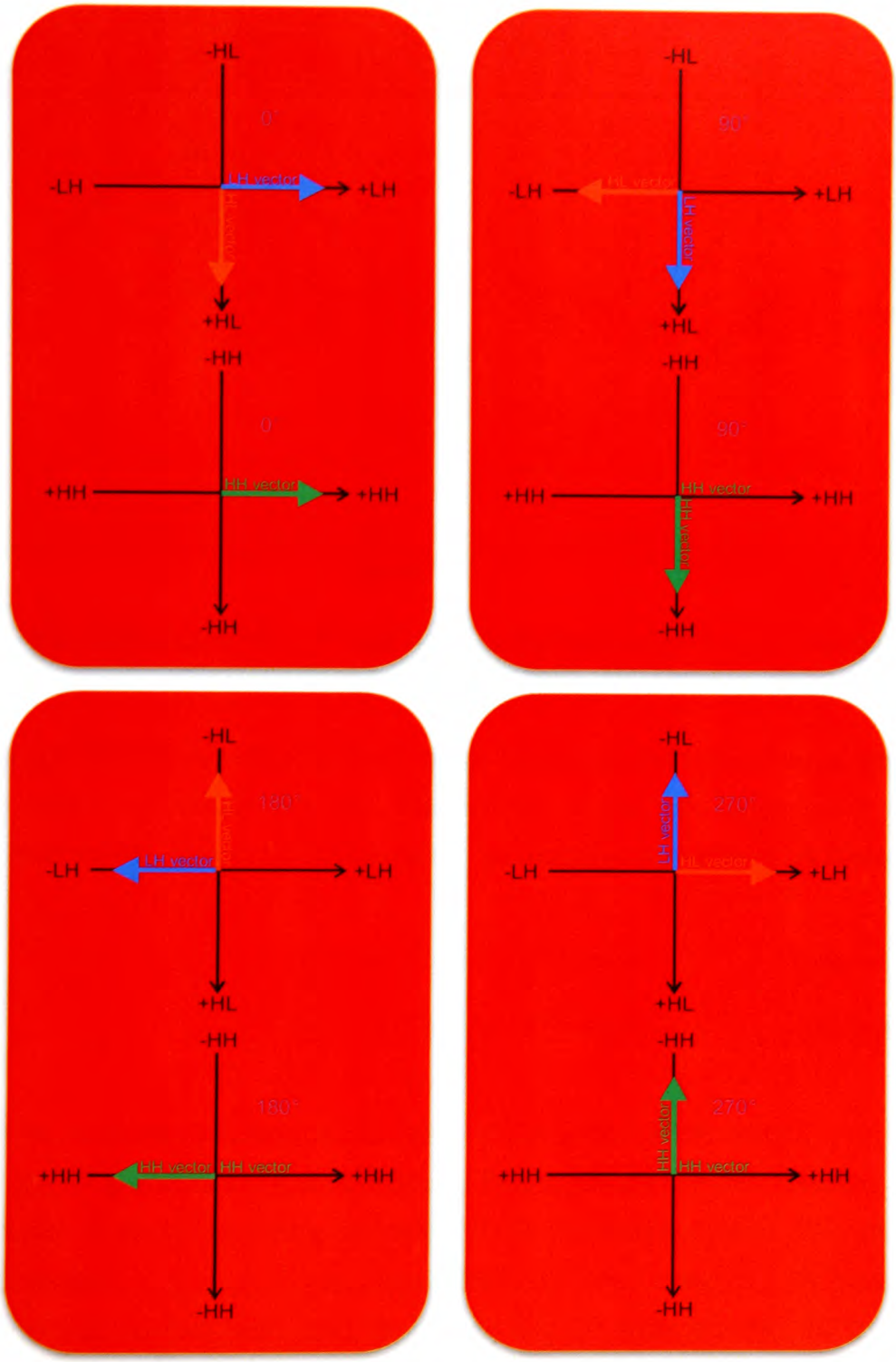


Figure 4.2: Sign changes of subbands in relation to the isometric operations.

Figure 4.3 is a plot of the value of the $C_{xy,\theta}{}^{HL}$ through a full rotation, expressed as a function of $C_{xy}{}^{HL}$ and $C_{xy}{}^{LH}$. Interpolation of the coefficient values for the intermediate angles reveals the sinusoidal variation of the contribution of each of the two factors, to the final coefficient value, with a phase difference of $90°$ between them.

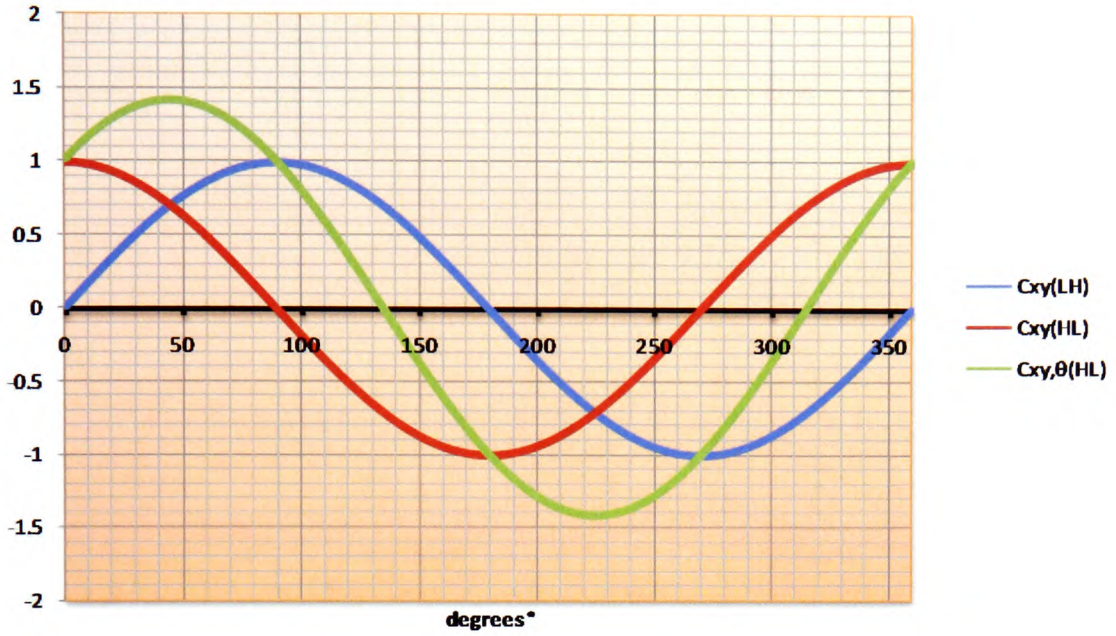


Figure 4.3: Contribution of C_{xy}^{HL} and C_{xy}^{LH} to the final value of $C_{xy,\theta}^{HL}$ throughout a full rotation cycle. Unity magnitude assumed for both coefficients.

Assuming that the contribution of each of the horizontal and vertical detail subbands indeed varies sinusoidally for the intermediate angles, the magnitudes of the coefficients for the HL and LH subbands can be modelled respectively as follows:

$$C_{xy,\theta}^{HL} = C_{xy}^{HL} \cos\theta + C_{xy}^{LH} \sin\theta \qquad \textbf{[eq 4-1]}$$

$$C_{xy,\theta}^{LH} = C_{xy}^{LH} \cos\theta + C_{xy}^{HL} \sin\theta \qquad \textbf{[eq 4-2]}$$

For the HH subband there are two differences; the period of the change in sign is $\pi/2$ or half that of the HL and LH subbands and the magnitude remains constant. Figure 4.4 is a plot of the values that a coefficient of the HH subband takes. The values for the intermediate angles are interpolated using a 3nd order trend line. By making the same assumptions that were used for the HL and LH subbands the HH coefficient can be modelled after the following equation:

$$C_{xy,\theta}^{HH} = C_{xy}^{HH} \cos 2\theta \qquad \textbf{[eq 4-3]}$$

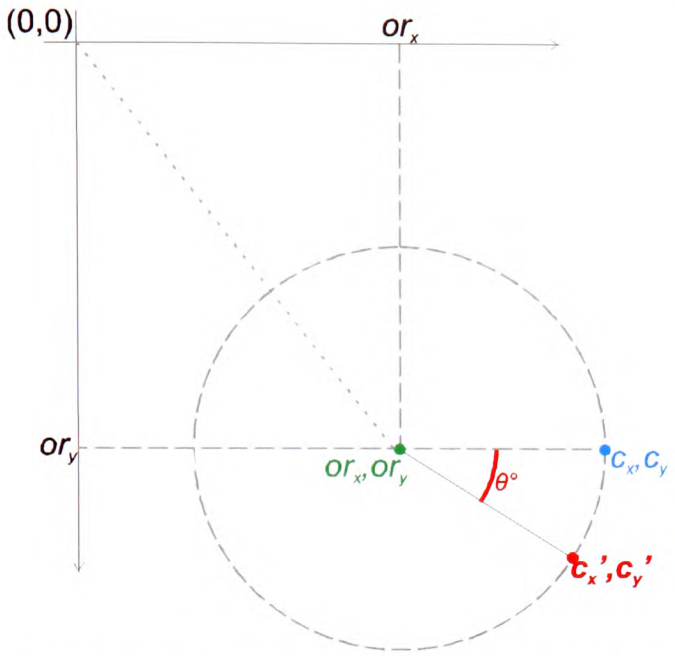


Figure 4.4: Calculation of rotated coordinates c_x', c_y' relative to origin or_x, or_y.

Note that the 2θ term inside the cosine reflects the halving of the period of the sign changes. We call the set of equations 4-1, 4-2 and 4-3 the *Wavelet Isometric Rotation Model.*

It must be noted that equations 4-1, 4-2 and 4-3 only describe the new value of the coefficient. However, when applying this model, the new coordinates of the coefficient must also be determined, as a final parameter. A coefficient with co-ordinates c_x, c_y can be rotated around the origin (or_x, or_y) using the following equations (see fig 4.4):

$$c_x' = (or_x + \cos(\theta) \times c_x) - (or_y + \sin(\theta) \times c_y) \qquad \textbf{[eq 4-4]}$$

$$c_y' = (or_x + \sin(\theta) \times c_x) + (or_y + \cos(\theta) \times c_y) \qquad \textbf{[eq 4-5]}$$

where θ is the rotation angle, and c_x', c_y' the new coordinates. Note that the c_x, c_y, c_x', c_y' coordinates are relative to the coordinates system with origin at or_x, or_y.

4.4 A conceptual approach to the Wavelet Isometric Operators assumption – enhancing the wavelet isometric model

As aforementioned, the detail subbands produced by the pyramidal wavelet decomposition are named after the directional information that they contain. Being a direct result of the high pass filtering process, the horizontal detail subband contains the high frequency energy components of the image in the horizontal direction. Similarly the vertical and diagonal detail subbands contain the high frequency energy components of the image in the vertical and diagonal directions respectively. The high frequency energy components describe the finer detail content of the image in a particular orientation. It is therefore valid to describe the content of the detail subbands as information about edges for a particular scale and orientation.

From the above, it follows that the magnitude of a detail subband coefficient can be considered as a metric of a specific high-frequency pattern (steep intensity change) at a specific location, orientation and scale. Therefore, it acts as an edge "detector". In general, higher coefficient magnitude means a sharper edge and a lower coefficient magnitude means a softer edge. Concluding, the three detail subbands, horizontal, vertical and diagonal can be thought as a quantitative representation of the edge content of the image in the respective orientation/direction.

Let us consider the case of a single rotating edge, as it moves within the wavelet decomposition for different angles of image rotation. As the edge rotates, one expects that the value of its horizontal, vertical and diagonal components (represented by the detail subbands in this case) will vary periodically. This was illustrated mathematically in the previous section by the wavelet isometric operators, albeit only for specific angles. In Figure 4.5 an edge is depicted as a vector with projections on the horizontal, the vertical and the diagonal axis (Figure 4.5a).

Consider a simple case of a sharp edge where the intensity goes from minimum to maximum and immediately back to minimum. A 360 degrees rotated version of such an edge in an image can be modelled by a circle with 1-pixel-wide periphery over solid background with maximum contrast.

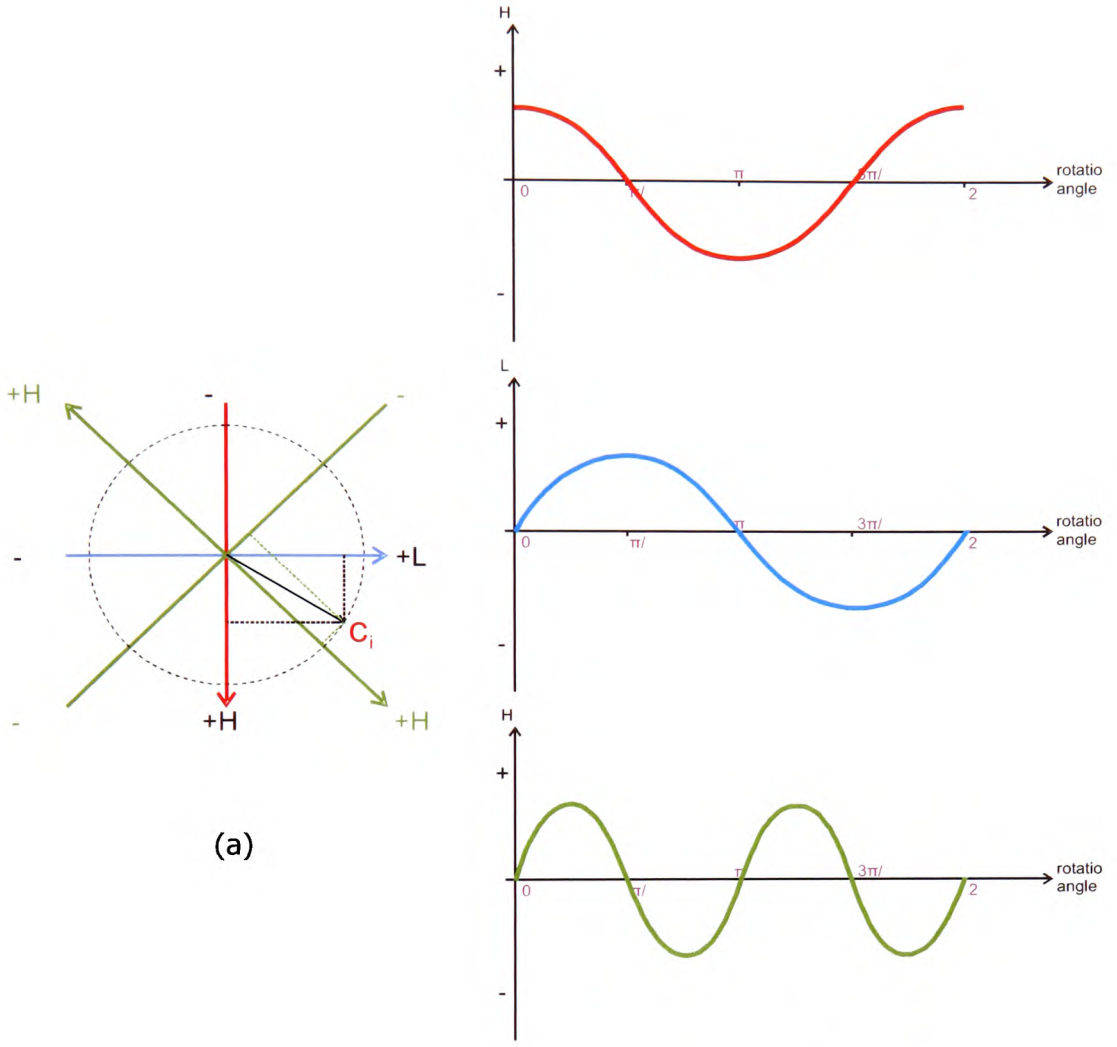


Figure 4.5: a) A rotating edge as a vector – conceptual model b) plot of the values of the projections on the vertical, horizontal and diagonal detail subbands.

Figure 4.6 shows the wavelet decomposition of such a circle. A black one pixel-wide circle is plotted against white background. The image is transformed using the Daubechies 5,3 filter introduced in Chapter 3. Absolute values instead of normal (signed) values have been used for visualization of the subbands to enhance visibility and clarity of the figure. As a result of this normalization, a brighter spot in a subband designates a coefficient with higher magnitude and a darker spot a coefficient with lower magnitude. The intensity, and therefore the energy content of the edge, varies in a manner very close to the suggested model, as we move around the perimeter of the circle.

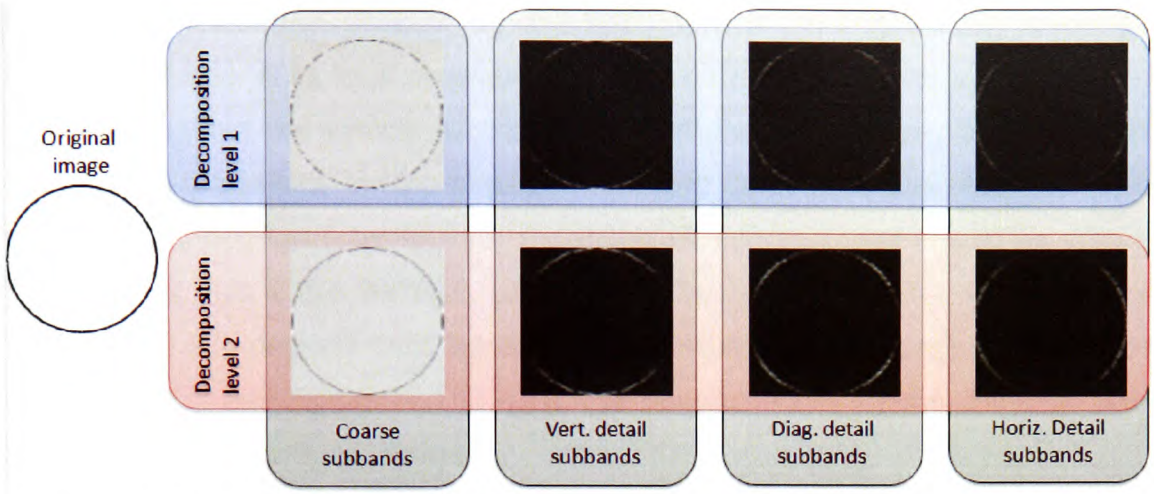


Figure 4.6: 2 level DWT decomposition of a circle – subband coefficients are visualized as absolute values for clarity.

The plots of the projections on the vertical, horizontal and diagonal detail subbands (fig 4.5b) resemble closely the model described in the previous section with one exception; that is, the Wavelet Isometric Rotation Model does not take into account any correlation between the values of vertical and the diagonal detail components.

The term vertical is used here to describe both the horizontal and vertical components. The problem introduced by this omission can be better illustrated by considering the example of a $45°$ rotation of the edge in Figure 4.5. Using equations 4-1, 4-2 and 4-3 for a $45°$ rotation yields:

$$\sin 45° = \cos 45° \approx 0.707$$

$$\cos(2x45°) = 0$$

(eq. 4-1)
$$C^{HL}_{xy,\theta} = C^{HL}_{xy} \cos 45° + C^{LH}_{xy} \sin 45° = 0.707\, C^{HL}_{xy} + 0.707\, C^{LH}_{xy}$$

(eq. 4-2)
$$C^{LH}_{xy,\theta} = C^{LH}_{xy} \cos 45° + C^{HL}_{xy} \sin 45° = 0.707\, C^{LH}_{xy} + 0.707\, C^{HL}_{xy}$$

$$C^{HH}_{xy,\theta} = C^{HH}_{xy} \cos 90° = 0$$

Therefore,

$$C^{HL}_{xy,\theta} = C^{LH}_{xy,\theta} = 0.707\, C^{HL}_{xy} + 0.707\, C^{LH}_{xy}$$

$$C^{HH}_{xy,\theta} = 0$$

However, based on the model of Figure 4.5 the value of the diagonal component $C_{xy}{}^{\theta}$ at $90°$ should be at its local maximum! In order to incorporate this into the model, information about the vertical subbands is included in the diagonal detail subband. To preserve the validity for the already established isometric model at angles of 90, 180 and 270 degrees (see fig.4.1); the newly introduced factor should be zero at those points. This is true because, according to the original isometric model, $C_{xy}{}^{\theta}$ at 90, 180 and 270 degrees must be equal in magnitude to the original HH coefficient (see table 4.1 and Figures 4.1 & 4.2). The revised model for the diagonal detail coefficient is therefore, as follows:

$$C_{xy,\theta}^{HH} = C_{xy}^{HH} \cos 2\theta + C_{xy,\theta}^{HL} \sin 2\theta + C_{xy,\theta}^{HL} \sin 2\theta \qquad \textbf{[eq 4-6]}$$

Note that, for 0, 90, 180 and 270 degrees the introduced sinusoidal factors are zero, and therefore they do not invalidate the rules of the original model. Table 4.2 lists the complete set of equations that comprise the revised wavelet isometry model. Equations 4-8 and 4-9 are repeated for completeness.

Table 4.2: Wavelet Isometry Model

Horizontal Subband coefficients	$C_{xy,\theta}^{HL} = C_{xy}^{HL} \cos\theta + C_{xy}^{LH} \sin\theta$	**[eq 4-1]**
Vertical Subband coefficients	$C_{xy,\theta}^{LH} = C_{xy}^{LH} \cos\theta + C_{xy}^{HL} \sin\theta$	**[eq 4-2]**
Diagonal Subband coefficients	$C_{xy,\theta}^{HH} = C_{xy}^{HH} \cos 2\theta + C_{xy,\theta}^{LH} \sin 2\theta + C_{xy,\theta}^{HL} \sin 2\theta$	**[eq 4-6]**
new x coordinates	$c_x' = (or_x + \cos(\theta) * c_x) - (or_y + \sin(\theta) * c_y)$	**[eq 4-4]**
new y coordinates	$c_y' = (or_x + \sin(\theta) * c_x) + (or_y + \cos(\theta) * c_y)$	**[eq 4-5]**

Since the Isometry Operators shown in table 4.1 provide a quick means of performing per-quadrant rotations (90°, 180° and 270°), it is only necessary to calculate the coefficients for angles between 0° and 90°. The Isometry Operators of table 4.1 can then be applied in order to calculate the coefficients for the remaining angles.

4.5 Achieving rotation invariance

4.5.1 Subband statistics through the Wavelet Isometric Rotation Model.

The energy or mean deviation of the DWT subbands is commonly used for the derivation of texture feature information (See Sections 3.2 and 2.7). It has been proven (Van de Wouwer, Scheunders *et al.* 1999b) that since G in equations 3-2 to 3-4 is a high-pass filter, the mean of the wavelet detail coefficients equals zero. Consequently, the variance equals the mean deviation of the coefficients and can be used as an equivalent measure of subband energy. The variance can therefore be used to describe texture properties of an image.

It has been observed (Brady and Xie 1996; Lu 1997) that the DWT tree is not invariant to affine transformations of the image. Affine transformations include linear transformations (rotation, scaling or shear) and translations (or "shifts"). Here the Isometry Operators are used in order to overcome this issue and achieve rotation invariance.

Two statistical measures are employed, namely, the subband energy histogram and the variance, in order to construct two rotation invariant texture descriptors. The concept behind the proposed techniques is that the relevant statistic is calculated using the available coefficient data. Then, with the aid of the Wavelet Isometry Model, the statistics are recalculated for a number of predefined angles, and hence a set of features is generated for rotation-invariant texture characterization. The choice of these particular statistical methods is based on their proven efficiency in feature characterization applications and the simple implementation of the experimental platforms.

4.5.1.1 Energy Histogram

The parameters that must be defined when calculating the subband energy histogram are: the number of bins (horizontal axis resolution) and, the quantization of the bin contents (vertical axis resolution). Both parameters affect the amount of information enclosed and the resulting size of the histogram.

The texture descriptor is created by calculating the histograms of all detail subbands of the DWT. For each subband, a histogram of 256 bins is constructed. The wavelet coefficients are first quantized to 256 levels and then each coefficient is assigned to the respective histogram bin. Two versions of the texture descriptor are proposed with respect to their size; the full histogram descriptor and the quantized histogram descriptor. For the construction of the full texture descriptor, the 256x256 histogram is calculated for 24 angles (0° to 360° using a 15° step).

For the quantized descriptor, the histogram is thresholded in order to create a binary signature of 256 bits = 32 bytes (equal to the number of bins). The choice of the threshold is very important as a high threshold would result in a "weak" feature (a bin would be very difficult to qualify) and a low threshold would result in a saturated feature (most bins would qualify in all cases). Both situations would affect negatively the discrimination ability of the feature. Therefore, the threshold is chosen so that 30% to 50% of the bins qualify. Since the values of the coefficients depend on the filter chosen for the decomposition, the same filter is used by both test and query images in order to maintain consistency.

After calculating the initial histogram, the Wavelet Isometry Model is used in order to calculate the coefficient values, and hence the histograms for other angles. Note that, since the aim is not to perform a full inverse DWT, the positions of the coefficients are of no concern. This greatly reduces the amount of processing required compared to reconstructing the entire rotated decomposition tree. Figure 4.7 is a schematic of the process followed for one angle at one level of decomposition.

The feature contains 23 signatures for each equally spaced successive angle. Considering that, 3 × 32 bytes are required per angle and per level of decomposition, the signature size for n levels of decomposition is (2.156 × n) Kbytes.

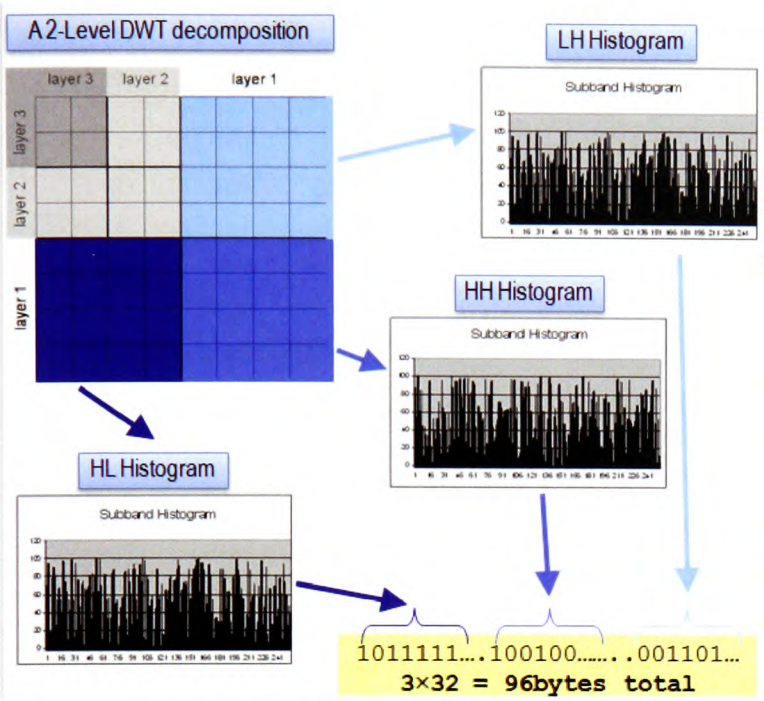


Figure 4.7: Example of the process for the calculation of a feature signature at one specific angle and one decomposition level.

Rectangles of different shades of blue represent the LL, HL and LH subbands for a specific decomposition level.

Note that, this process must be repeated for all required levels of decomposition. Then, the histograms hence the signatures are recalculated for each angle at 15° intervals.

The technique is flexible in that database image signatures can be calculated prior to the query but, there is no need to calculate rotated versions for the whole database. Instead the feature signature of the query image is calculated for all angles once, during query process initialization. Then, each test image signature is compared against all the angle signatures of the query image.

Algorithm 4.1: Calculation of histogram-based rotation invariant texture descriptor

Database Pre-processing stage:

a) For each image I with m DWT detail subbands inside the database calculate (m) energy histograms with 256 bins and 256 quantization levels.

b) Let B_k, $k=1,\ldots,256$ be the bins of the energy histogram: Using a threshold T_l create m 256 bit long binary signatures with respect to the following rule:

$$bit_k = \begin{cases} 1, B_k >= T_l \\ 0, B_k < T_l \end{cases}, k = 1,\ldots,256 \qquad \textbf{[eq 4-7]}$$

Classification Pre-processing stage:

a) For query image Q with m DWT detail subbands calculate (m) energy histograms with 256 bins and 256 quantization levels.

b) Let B_k, $k=1,...,256$ be the bins of the energy histogram: Using a threshold T_l create m 256 bit long binary signatures with respect to equation 4-11.
c) Using the Wavelet Isometry Model calculate the histograms for angles between 15° to 345° (in 15° steps) and repeat steps (a) and (b) for each angle.

4.5.1.2 Variance

Let c_n the wavelet coefficient of a subband containing n coefficients, and m the mean of the coefficients of the specific subband. Then, the variance is given by:

$$variance = \frac{(c_0 - m)^2 + (c_1 - m)^2 + ... + (c_n - m)^2}{n-1} \quad \textbf{[eq 4-8]}$$

An example of the use of variance as a texture feature in wavelets domain can be found in (Simoncelli and Portilla 1998). Here, in order to construct the rotation invariant texture descriptor, the Wavelet Isometry Model (table 4.2) will be incorporated into equation 4-8. The first step is to replace the coefficient values (c_0, c_1, c_2.....) by the respective equations of the Wavelet Isometry Model. The other element of equation 4-8 that is directly affected by the values of the coefficients is the mean, which is defined as:

$$m = \frac{\sum_{j=1}^{n} c_j}{n} \quad \textbf{[eq 4-9]}$$

Therefore, it is also necessary to recalculate the mean for each desired rotation angle. Note that this can also be achieved without recalculation of the full DWT tree and only by replacing the coefficient values with the respective Wavelet Isometry Model equations.

The signature vector for a specific angle is equal in size to the number of detail subbands. Here follows the algorithm for variance-based texture extraction incorporating the Wavelet Isometry Model of table 4.2:

Algorithm 4.2: Calculation of variance-based rotation invariant texture descriptor at angle $\theta°$

Database Pre-processing stage:

a) For each image I_1 with m DWT detail subbands inside the database calculate (m) variances using equation 4-8.
b) The calculated variances are rounded up to 3 decimal digits in order to form an m-elements long feature vector.

Classification Pre-processing stage:

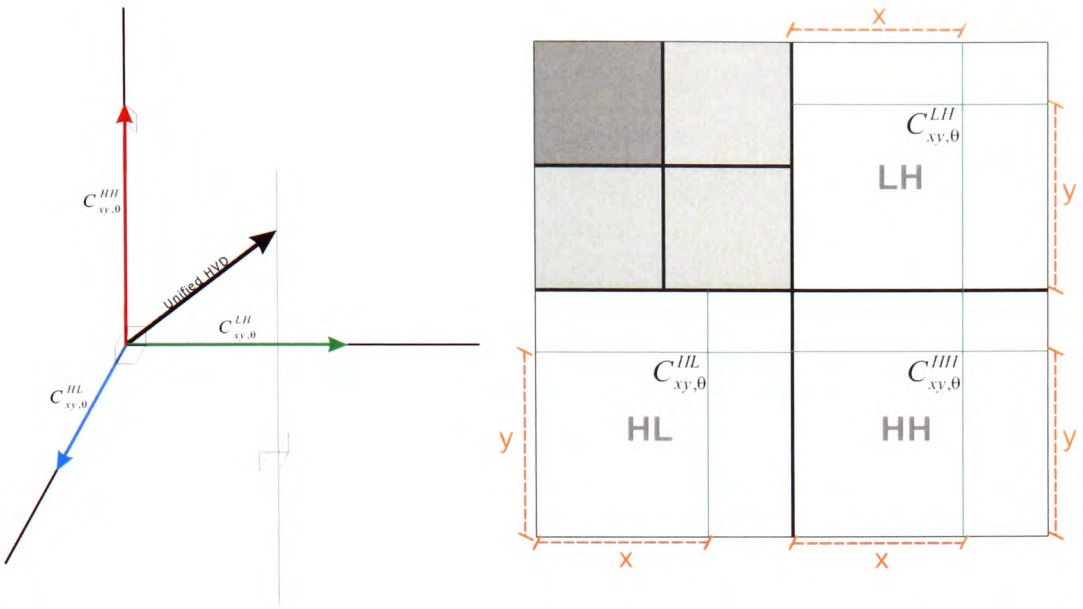
a) For query image Q with m DWT detail subbands calculate (m) variances using equation 4-8.
b) The calculated variances are rounded up to 3 decimal digits in order to form an m-elements long feature vector for the specific angle.
c) Using the Wavelet Isometry Model in equations 4-9 and 4-8 repeat steps (a) and (b) accordingly in order to calculate the mean and the respective subband variations for angles between 15° to 345° (in 15° steps).

4.5.2 The Unified HVD Space

The Wavelet Isometry Model, which was introduced in Section 4.4, provides a means of correlating the coefficients between subbands with different orientations at the same level of decomposition. Disregarding the actual displacement of a rotated coefficient, the Wavelet Isometry Model provides a system of three equations, where the rotation angle $\theta°$ is the only variable. The three vectors produced by the Wavelet Isometry Model equations can be combined in a single 3D vector as shown in Figure 4.8(a). For convenience, we call the resulting three-dimensional vector the Unified HVD vector, and the respective three-dimensional locus the Unified HVD space. One Unified HVD vector, therefore, corresponds to a set of three coefficients at the same relevant x,y coordinates in the LH, HL and HH subbands (see Figure 4.8(b)).

The $[X,Y,Z]$ coordinates that define a Unified HVD vector are provided by the Wavelet Isometry Model as follows:

$$\text{Unified HVD vector} = [X,Y,Z] = \left[C_{xy,\theta}^{LH}, C_{xy,\theta}^{HL}, C_{xy,\theta}^{HH}\right] \quad \textbf{[eq 4-10]}$$



(a) (b)

Figure 4.8: a) The Unified HVD vector b) Relative location of Unified HVD vector components in DWT decomposition.

In order to construct a texture feature descriptor based on the Unified HVD vector, the vectors are calculated for the three subbands thereby forming a feature space equal in size to 1/3 of the original subbands (one vector per three coefficients). The histogram of the resulting vector representation is the texture descriptor of the image. In order to facilitate the construction of the histogram, only the magnitudes of the Unified HVD vectors are considered, reducing in this manner the size of the produced data and the required calculations. A granularity of 256 bins and 256 quantization steps per bin is used. Since a different Unified HVD vector representation is required for each DWT level of decomposition, the size of the signature depends on how many levels of decomposition are going to be taken into consideration.

Algorithm 4.3: Calculation of Unified HVD vector texture descriptor

Database Pre-processing stage:

- For each image I inside the database calculate Unified HVD histograms for each available level of decomposition using equation 4.10 with 256 bins and 256 quantization levels.

Classification Pre-processing stage:

a) For query image Q, calculate the Unified HVD histograms for each available level of decomposition using equation 4.10 with 256 bins and 256 quantization levels.
b) Incorporating the Wavelet Isometry Model in equation 4-14, repeat step (a) accordingly in order to calculate the Unified HVD histograms for angles between 15° to 345° (in 15° steps).

4.6 Experimental setups

4.6.1 Datasets

The performance of the algorithms presented in the chapter is evaluated for image retrieval and classification applications. In order to incorporate the rotation invariance characteristic, the experiments were performed on a subset of the Outex Database, which contains samples from 319 textures (Ojala, Maenpaa et al. 2008). The Outex database was constructed under controlled illumination and spatial resolution conditions to ensure uniform imaging geometry. The Outex framework is described in detail in (Ojala, Maenpaa et al. 2002). The advantage of using this data set over those used in Chapter 3 is that rotated versions of the images have been acquired during capture at the controlled environment described above. Thereby the rotated test set is closer to a real life input for the systems rather than using

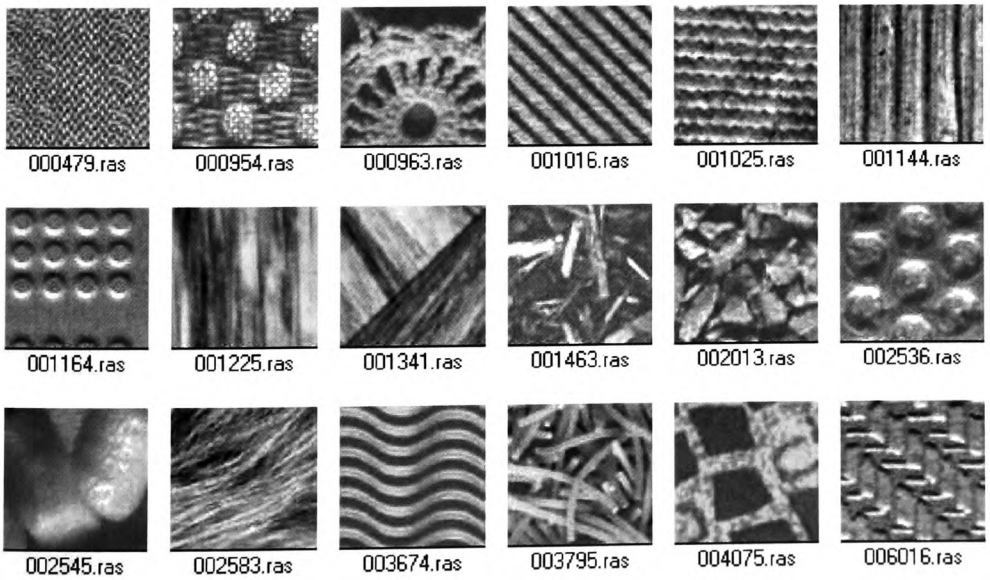


Figure 4.9: Database D - Outex retrieval samples.

artificially generated rotated versions of the images described in Chapter 3. Furthermore using this data-set enables future comparison of this work with existing research.

4.6.1.1 Generic Image Rotation Invariant Indexing and retrieval (Database D)

The retrieval experiments were performed on a database of 6380, 8bpp, grey scale images, covering all 319 textures of Outex at angles 0°, 15°, 30°, 45°, 60°, 75° and 90°. The set included 2 images for each angle per texture (total 4466) and an addition of 6 non-rotated images per texture (total 1914) in order to be able to perform comparative rotated versus non-rotated experiments. The primary reason for selecting this database for the retrieval experiments is the very broad thematology of the content, therefore offering a broad image domain whilst including samples at various orientations. The size of the images is fixed to 128 by 128 pixels. Figure 4.9 shows selected examples from Database D.

4.6.1.2 Rotation Invariant Classification (Database E)

The classification experiments were performed on a database of 1920, 8bpp, grey scale images, covering 16 different Brodatz (Brodatz 1966) textures at angles 0°, 5°, 10°, 15°, 30°, 45°, 60°, 75° and 90°. The set included 10 different images for each angle per texture class (total 1440) and an addition of 30 non-rotated images per class (total 480) in order to be able to perform comparative rotated versus non-rotated experiments. Figure 4.10 shows the 16 classes of Database E. The database has been selected here as a widely adopted test-bed for texture analysis evaluation (see Section 1.6.4 for examples). Additionally Database E contains both more angles per texture class and more images per angle thereby providing a richer dataset for training the classifier. Therefore Database E was chosen as more suitable for the classification experiments whereas Database D, which focuses on the variety of content, was chosen for retrieval.

4.6.2 Evaluation of features & implementation specifics

4.6.2.1 Generic Image Indexing and Retrieval Accuracy

As it was discussed in Section 3.5.2.1 and 1.6.4, the commonly used retrieval evaluation metrics of *precision, recall* and their derivatives rely on strictly defined

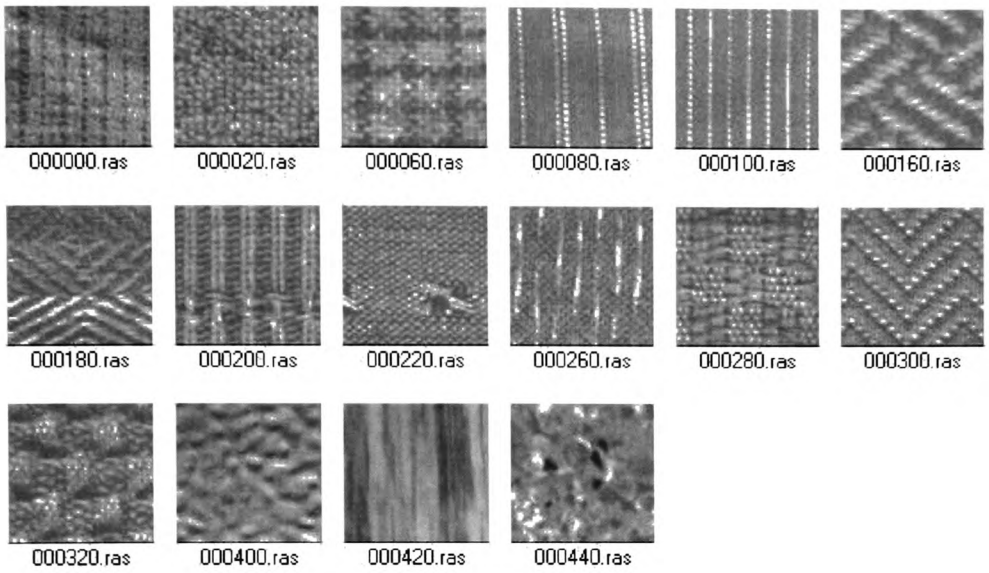


Figure 4.10: Database E Brodatz texture classes samples.

delineation between different image categories. The reason is that when calculating either *precision* or *recall* image pairs can only be characterized as similar or dissimilar. Database D contains 319 labelled categories of images. This categorization provides a ground truth for establishing image similarity and as a result it can be used in conjunction with any of the aforementioned metrics for determining the accuracy of the systems. Although this constitutes a solid condition for identifying the members of the specific class it is still possible that images sourced from different scenes may have similar visual characteristics.

4.6.2.2 Classification Accuracy

Following the discussions of Sections 1.6 and 3.5.2.2 the accuracy of the classification process is validated via the *Knn*-classifier (k=3) with majority voting and the leave-one-out technique. Section 3.5.2.2 elaborates on both techniques. Note that for the classification experiments the query image is tested for similarity against the pre-defined image classes of Database E.

4.7 Empirical analysis – Algorithm response with incomplete set of DWT data

The processing required for both retrieval and classification is proportional to the actual volume of data. It is therefore desirable to be able of extracting all necessary feature information from a smaller data set without compromising accuracy.

Furthermore, considering the compression and feature extraction problem jointly, the algorithms may have to cope with images that are already compressed using a DWT decomposition tree with a specific layer depth. Another option would be to further process the DWT tree in order to obtain more layers of decomposition, provided that this is allowed by the image resolution. However, this would "move" the feature extraction algorithm to the left of point C in figure 1.2, thereby disqualifying it from operating in "compressed domain". To this end, in this first part of the empirical analysis, the effect that different levels of available wavelet information have on the accuracy of the algorithms is investigated.

The first approach is to exclude one or more layers of decomposition from the calculation process. Starting from a single layer of decomposition (*i.e.* from finer detail) , additional (coarser) layers are added. This also provides an indication of the extent to which coarse image features and finer detail image features contribute to the discrimination ability of each method.

The second approach is applied after the calculation of the Significance Map. After establishing the importance of each subband, the least important subbands are excluded. Since it is unfeasible to pre-determine which layers contain the least important subbands, this approach can only be applied to methods that perform per-subband and not per-layer calculations. This analysis provides an indication of the extent to which subband importance contributes to the discrimination ability of each method.

4.7.1 Levels of DWT Decomposition

In this section, the effect of having images decomposed at different levels is studied. The non-quantized versions of all algorithms are used so that the maximum content information is available. Figure 4.11 depicts the results of the experiments.

Observation of the graph shows that the Unified HVD has a clear advantage other the Energy Histogram and the Variance-based descriptors. The Unified HVD demonstrates a relatively high level of accuracy with a single level of decomposition (84%), saturates at three levels of decomposition (96.2%) and shows a slight down

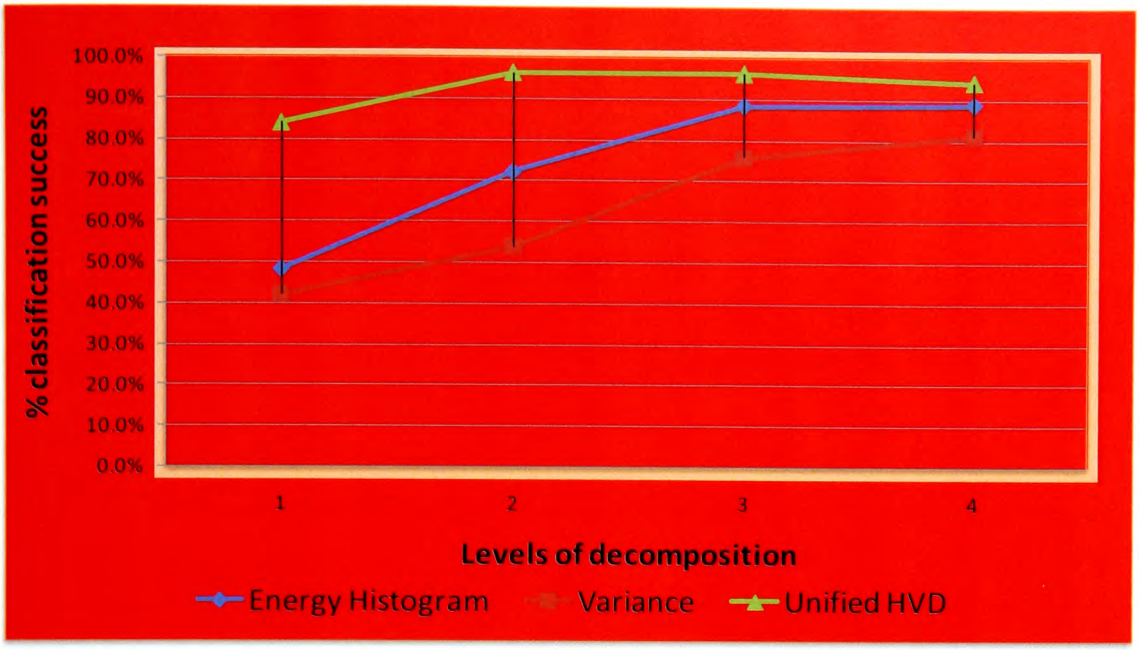


Figure 4.11: Effect of increasing levels of decomposition on the classification success of algorithms.

trend for four layers. This downwards turn can be attributed to the fact that the HVD operator depends more on high frequency information. As such, after the third layer the coarse subband information adds to the operator noise instead of additional information. Finally, the operator has overall the highest performance and shows the most stable response of the three. This stability can also be attributed to the fact that the operator depends on detail information and most of that becomes available at the first decomposition levels. On the opposite side the Variance feature is both the worse performing and requires all the decomposition levels in order to reach maximum accuracy (81.2%). A similar trend but with an average of 5% to 10% better accuracy is followed by the energy histogram. The ability of the Unified HVD operator to achieve high accuracy with 1-2 layers also favours applications where the image data are uncompressed thus requiring less DWT pre-processing.

4.7.2 Select subbands based on their importance

In this experiment, the relative importance of the 13 available subbands is first calculated. The energy histogram and variance features are calculated starting from the most important subband, gradually adding more subbands that are less important. The aim is to determine whether higher accuracy can be achieved by omitting the least important subbands of the DWT.

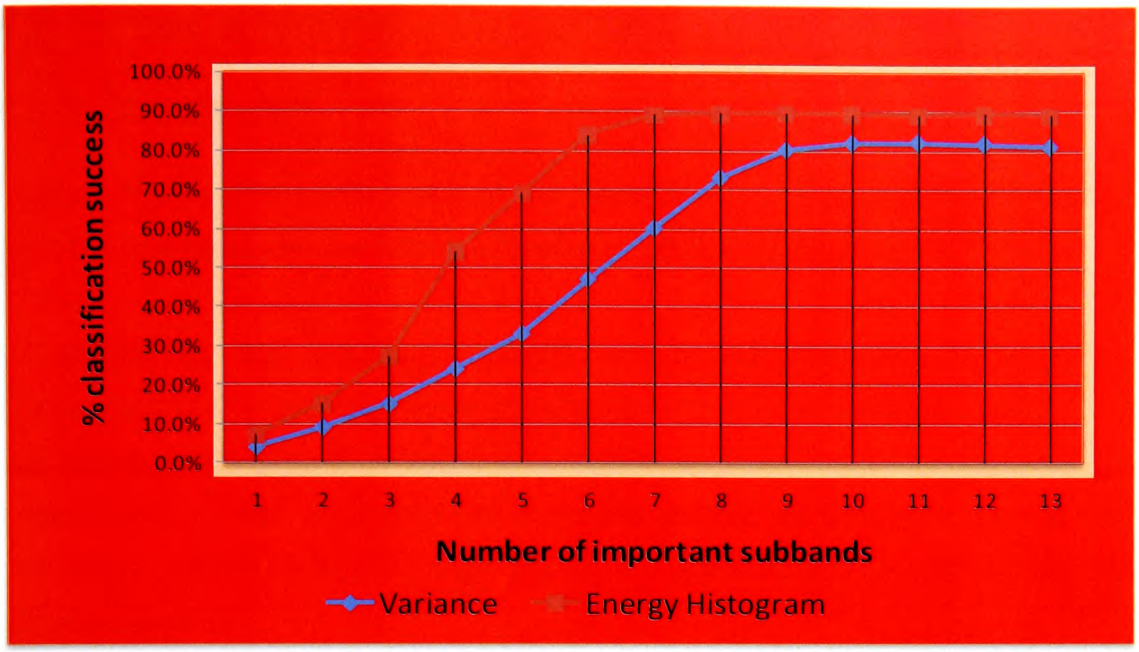


Figure 4.12: Effect of increasing number of important subbands on the classification success of algorithms.

The Unified HVD method is not evaluated against the number of important subbands since it requires information from all HL, HH and LH subbands by design. The results of the experiments are shown in Figure 4.12. From the graph , it is can be seen that the energy histogram feature saturates when approximately 60% of subbands are used (89% classification accuracy). The Variance feature requires more information than energy histogram (approximately 75% of the available subbands) in order to reach maximum performance accordingly. In both cases the algorithms show slightly deteriorated performance when all available subbands are used. This is expected since the least important subbands contain by definition less energy, therefore less content information, and hence adding more "noise" to the sample. A general observation is that, in a typical compression codec where subband importance can be easily determined at the decoder stage, savings of approximately 35-50% in data processing can be achieved by disregarding the least important subbands.

4.8 Empirical analysis – Classification and retrieval

4.8.1 Classification

Texture classification performance of the algorithms is evaluated using Database E. Both the quantized and the non-quantized versions of the algorithms are tested.

In addition to the individual Variance and Energy Histogram techniques, a combination of the two was also implemented for the classification experiment. The aim of this combination is to investigate whether the accuracy of the two algorithms is cumulative or not. The combined feature takes into account both metrics when comparing and classifying an image. During the experiments the two features were equally weighted.

Table 4.3: Classification performance of proposed algorithms. Results expressed in % success.

	Non-rotated samples		**Rotated samples**	
	non-quantized	**quantized**	**non-quantized**	**quantized**
Variance	77.0%	73.5%	62.0%	59.5%
Energy Histogram	89.0%	82.0%	85.7%	79.5%
Energy + Variance	91.7%	84.0%	88.0%	83.0%
Unified HVD	96.0%	88.5%	95.7%	87.0%

The classification results are shown in table 4.3. The performance of the three algorithms is in line with the previous examples with Unified HVD offering the best overall accuracy. As expected, the increased difficulty of the rotated samples affected negatively the accuracy of all algorithms. Interestingly, the joined energy and variance feature outperforms both individual feature responses. This means that the two features are partially uncorrelated and can be used in parallel to improve accuracy. With regards to the quantized versions of the techniques, quantization appears to reduce the accuracy by approximately 3-6%. The most invariant feature is Variance albeit at a considerably lower average accuracy than the rest of the algorithms. Finally, the Unified HVD operator shows a high level of stability in the performance between the rotated and un-rotated experiments with less than 1% drop.

4.8.2 Image Retrieval

Here the algorithms are evaluated on image retrieval using database D. The L_2-distance between two feature vectors is used to measure image similarity. The selected query image was removed from the database during processing.

The introduction of the much more diverse data set and the increased inherent difficulty of the retrieval problem are evident. The results are consistently lower than the classification experiments on Database E. The increased entropy of the dataset and the evaluation method appear to have affected the performance of the joined energy and variance operator. Here the difference between the individual operators and the joined operator is marginal, thus indicating selectivity on the data set. The gap between non-quantized and quantized versions is also wider for retrieval which varies between 5% and 10%. The Unified HVD is again the best performing feature overall.

Table 4.4: Retrieval performance of proposed algorithms. Results expressed in % success.

	Rotated samples		**Non-rotated samples**	
	Automated		**Automated**	
	non-quantized	**quantized**	**non-quantized**	**quantized**
Variance	39.4%	36.7%	45.4%	42.7%
Energy Histogram	47.0%	40.2%	57.0%	55.2%
Energy + Variance	50.0%	41.9%	62.0%	53.1%
Unified HVD	56.0%	47.3%	62.0%	58.3%

4.9 Conclusions

In this chapter, the issue of narrowing the sensory gap is addressed and in particular how to improve the performance of rotation invariant texture characterization in wavelets domain. The motivation of this work was based on the facts that: a) the research on orthogonal DWT-based techniques which can be easily applied in compressed domain is scarce and b) there are pixel-domain techniques which still

outperform much more sophisticated separable wavelet-based schemes. The methods that are described in this chapter aimed to address those issues.

Building upon the observation by (Li and Jay Kuo 1999) that rotating the orthogonal wavelet subbands at specific orientations can be performed using only a few operations, a novel technique for mapping wavelet coefficient data on a rotated locus is introduced. The technique is built around an approximation model, called Wavelet Isometric Operators, aiming to predict the variations that the DWT coefficients undergo, as an image gets rotated. The model was employed in the design of two rotation-invariant texture descriptors. The first approach is an augmented rotation invariant form of subband statistics, namely the variance and the energy histograms.

Energy Histogram and Variance techniques estimate the respective subband statistics for the query image at various angles using the Wavelet Isometric Operator, thus constructing a multi-angle signature. The second approach is a new algorithm called the Unified HVD operator. For the creation of the texture operator, the Unified HVD directly applies the Wavelet Isometric Model to sets of multi-directional coefficient information. The Unified HVD has similar characteristics to the autoregressive occurrence model employed in the LWCP algorithm in the previous chapter albeit applied in wavelets domain in a very different way. The operator exploits the convolution property of the wavelet filter during decomposition in order to obtain information about neighbouring locations instead of using intra-subband patterns.

DIFFERENT LEVELS OF DWT DECOMPOSITION

The ability of the algorithms to give accurate results with variable levels of DWT decomposition was tested. Both statistical analysis methods require at least 3 levels of decomposition in order to provide accurate results. This does not come as a surprise since, both energy histogram and variance-based methods describe features using a single number per subband. With less than three levels of decomposition, the available subbands are inadequate in order to provide the required granularity for the system to have sufficient discrimination ability.

The Unified HVD implementation, which focuses mostly on intra-subband micro-features, reaches maximum accuracy with only 2 levels of decomposition.

Interestingly, the performance of the Unified HVD algorithm deteriorates when more levels of DWT decomposition are introduced. One possible reason is that, excessive content information about different resolutions is introduced. Since the system does not have provision for proper manipulation of such information, multi-resolution data are translated as noise. A multi-resolution version of the algorithm could probably remedy the problem.

FILTERING SUBBANDS BASED ON THEIR IMPORTANCE

The ability of the two statistical methods to accurately characterize texture without using the full DWT tree data is tested by screening out subbands of lower importance. For four levels of decomposition, both Energy Histogram and Variance saturate when using approximately 60-75% of the total number of subbands. It is worth noting that, by adding the subbands with the lowest importance, the accuracy of both techniques slightly deteriorates since, the additional information is mostly translated by the feature extraction algorithms as noise.

CLASSIFICATION AND RETRIEVAL

The classification accuracy of the three implementations was tested both on un-rotated and rotated datasets. Additionally, an implementation that combines energy histogram and variance on a single feature is introduced.

The algorithms represent different trade-offs between accuracy and storage/processing requirements. Specifically, Energy Histogram and Variance algorithms consider global statistics of the DWT. Therefore, these two algorithms have lower processing cost but perform worse than Unified HVD. Experimental results from a combined Energy Histogram-Variance feature indicate performance which is closer to that of Unified HVD.

Unified HVD is the best performing algorithm for both retrieval and classification. The Unified HVD operator can achieve good accuracy with only 1 or 2 layers of decomposition, thus it also favours applications where the data are not compressed. Introducing the rotated data set reduces the accuracy of all implementations by 5-7%. A general observation is that the retrieval results obtained for the rotation

invariant methods indicate that, the algorithms need further refinement in order to achieve comparable accuracy to existing non-rotation-invariant systems.

In environments where the importance of subbands is easily derived, for example, at the coding stage of a codec, importance can be used in order to filter out 35% - 50% of the data resulting in proportional processing savings.

Chapter 5: Improving the Speed of CBIR

5.1 Abstract

In this chapter, a number of techniques are presented which are designed to reduce the processing time required by a CBIR system, in order to perform a query, when all user defined parameters (e.g. query image, feature selection, feature parameters) are given. Two of the presented techniques improve the feature signature distance calculation process, and the third technique extracts feature content information directly from the compressed bit stream.

5.2 Introduction

While existing systems developed by the research community addressed some key issues of retrieval accuracy, a common issue is that the process of search is lengthy and time consuming, since all systems involve comparisons being carried out between the query image and all images inside the databases. Very large image databases built up to include millions of images and prime frames from video clips are already commonplace. This prompts a challenge to the area of content-based image retrieval, where content description and matching always requires more time and computing cost compared to conventional text-based retrieval. Therefore, maintaining a high image throughput is absolutely essential for all content-based image retrieval techniques.

This chapter presents a number of techniques addressing two of the original aims of this research, which are: a) to suggest methods for improving the speed of CBIR/CBIC systems, and b) to extract features directly from the compressed bit-stream. When designing for query speed improvement, different approaches need to be taken for systems that perform *a priori* calculation of the feature signatures and systems that extract feature information each time a query is performed.

In CBIR systems which employ pre-calculation of the feature signatures, the speed of a query is determined by the signature matching process and the size of the database. As many indexing keys are constructed in the form of histograms or

vectors, the content-based search is essentially governed by a distance matching process, in which every element inside the histograms/vectors need to be compared to contribute to its final distance value.

As it was discussed in Section 2.8, existing research in the field of similarity measurement optimization is scarce. Representative approaches include (Berman and Shapiro 1999), which suggests bisection of the search space and (Wilson and Bayoumi 2003) which suggests a method for successive refinement of distance calculation by introducing an extra symbol in the bit-stream of EZW.

In this work the bisection paradigm is extended by presenting two methods that feature tri-section of the feature space: a) by setting both upper and lower boundaries, and b) by a morphological analysis of the feature histogram signatures. Furthermore, based on two observations on how the L_2-distance is applied in CBIR for vector comparison a method for computationally simplifying the calculation is presented.

In CBIR systems that perform per query calculation of all the features, the speed depends on both the feature extraction algorithm and the time required for decompression of the database images. When attempting to improve the speed of the query it is desirable to preserve the retrieval accuracy of the system. Therefore, in order to fully maintain the feature extraction characteristics, the only option for speed improvement is to focus on the decompression process. Ideally feature information should be extracted with little or no decompression of the images.

Another contribution that is presented in this chapter is building upon the conclusion of Chapters 1 and 2 that there is a gap in the research for on-the-fly feature extraction from wavelet compressed images. Using as the basis SPIHT (Said and Pearlman 1996a), a well-known compression algorithm, this chapter introduces a technique for extracting significance map-based features directly from the compressed stream. The proposed retrieval system exploits the arithmetic coding and bit arrangement characteristics of SPIHT in order to bypass completely the decompression stage, thus resulting in significant savings in processing time.

Compared to existing approaches, the method presented here performs feature extraction on the compressed bit-stream offering a considerable advantage over techniques which require reconstruction of the entire DWT tree before being able to do any processing. Very few similar SPIHT-based techniques exist which require modification of the original algorithm. On the contrary, the method presented here is used in conjunction with the original algorithm, making it compatible with existing deployments and hardware implementations. Also, it does not suffer from the overhead of Tier-2 coding of the scarce JPEG2000-based techniques which do not require reconstruction of the DWT tree. In fact, the technique can be applied on the bit-stream without any intervention or processing from the SPIHT decoder. Finally, the technique can be used for successive refinement of distance calculation without the introduction of additional symbols of other techniques.

The technique is tested using the widely employed average subband energy as texture metric although it is directly extensible to any significant map-based descriptor such as those described in Chapter 3.

5.3 Pre-filtering type A; Faster L_2-distance calculation and morphological analysis of feature vectors

5.3.1 Computational Improvement of L_2-distance for CBIR applications

The Euclidian distance is used extensively in CBIR research due to its easy implementation and good accuracy, when it comes to feature matching. Given an image database, let q_k be the feature vector of the query image Q and i_k is the feature vector of a random image I inside the database. The Euclidian or L_2-distance between the query feature vector q_k and image texture vector i_k is:

$$d(I,Q) = \sum_k (i_k - q_k)^2$$

$$= \|i_k\|^2 + \|q_k\|^2 - 2\sum_k i_k q_k \qquad \textbf{[eq 5-1]}$$

where $k=1,2,...,m$ refers to the m individual elements of the feature vectors.

Let $i_{1,k}, i_{2,k}, i_{3,k} \ldots, i_{n,k}$ be respectively the feature vectors of all n images inside the database then equation 5-1 yields:

$$d(i_{1,k}, q_k) = \|i_{1,k}\|^2 + \|q_k\|^2 - 2\sum_k i_{1,k} q_k$$

$$d(i_{2,k}, q_k) = \|i_{2,k}\|^2 + \|q_k\|^2 - 2\sum_k i_{2,k} q_k$$

$$d(i_{3,k}, q_k) = \|i_{3,k}\|^2 + \|q_k\|^2 - 2\sum_k i_{3,k} q_k$$

...

...

$$d(i_{n,k}, q_k) = \|i_{n,k}\|^2 + \|q_k\|^2 - 2\sum_k i_{n,k} q_k$$

The query process therefore involves calculation of eq 5-1 for each feature vector and then sorting of the results in an ascending order. The top (smaller distance) results in the list represent the best matches for the specific query. If $i_{1,k}$, $i_{2,k}$ and $i_{3,k}$ are the feature vectors of the three best matches respectively, then the following inequality must be true:

$$d(i_{1,k}, q_k) \le d(i_{2,k}, q_k) \le d(i_{3,k}, q_k) \qquad \textbf{[eq 5-2]}$$

Two observations can be made from equations 5-1 and 5-2; firstly, for a specific query image, the $\|q_k\|$ term is constant in all distance calculations, hence it does not affect the order of the results, and secondly, the $\|i_{j,k}\|^2, j = 1,2,\ldots,n$ term is independent of the query image, and therefore remains constant between different queries.

These observations lead to the following conclusions:

a) Term $\|i_{j,k}\|^2$ need not be calculated for each query. It can be pre-calculated for each image and stored in a separate database.

b) Term $\|q_k\|$ can be omitted from the distance calculation process without affecting the results, and thus the accuracy of the retrieval process.

Therefore, the following simplified version of eq 5-1 can be used equivalently:

$$d'(i_k, q_k) = \|i_k\|^2 - 2\sum_k i_k q_k \qquad \textbf{[eq 5-3]}$$

5.3.2 Pre-filtering by morphological analysis of feature vectors

Most feature signatures generated by CBIR systems are in the form of histograms or vectors (See discussion of Section 2.8). Presuming that these histograms offer an accurate representation of the content of the image, it follows that a comparison between the histograms is equivalent to comparing the images with respect to the specific feature. For instance, in the case of the texture feature, two images with similar texture content will have more similar texture feature histograms than two images with dissimilar texture content.

The L*n*-distance has been broadly used as a metric of similarity between two histograms for image retrieval purposes. The signature histograms can also be viewed as two dimensional shapes (see Figure 5.1)

It is therefore valid to assert that, shape similarity evaluation techniques can be used in order to measure similarity between histograms. In this work, a simple morphological analysis employing the normalized moment of inertia (Alt 1962; Zusne 1965) of an area is used in order to pre-filter the feature histograms .

The moment of inertia has the advantage that it can be pre-calculated for all images in the database. The pre-filtering process is used in order to narrow the search space to those images with the most similar histograms, using *n*-order moments as

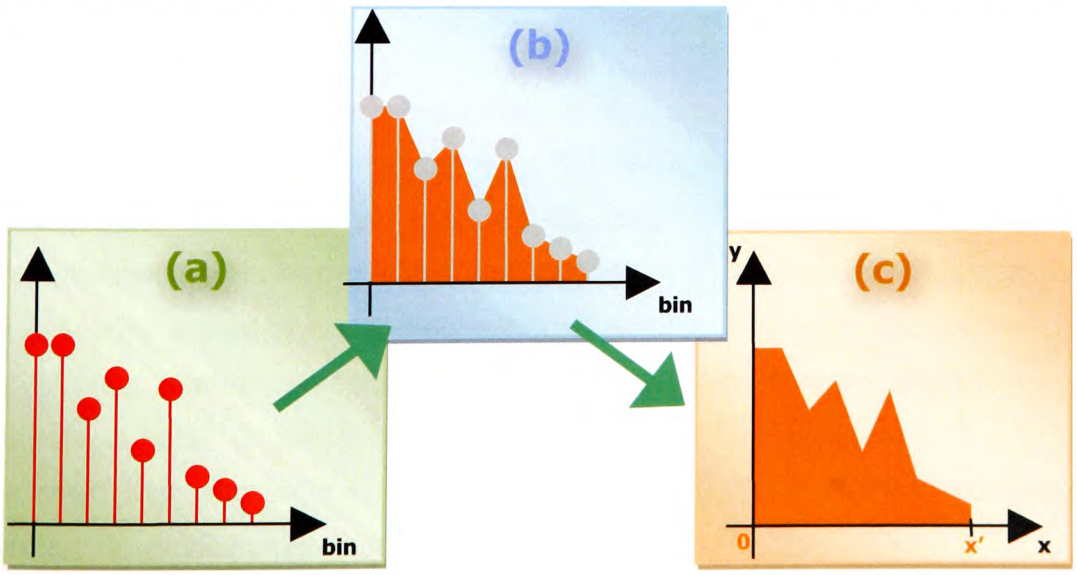


Figure 5.1: conceptual representation of a histogram as a 2d shape.

the criterion. In typical shape and pattern matching applications more than one moment are used for characterization. Here, it is important that the histogram is represented by a single value enabling thus the sorting of the image database with respect to this value.

Consider the area defined by the 2-D shape depicted in Figure 5.1(c), expressed as a one dimensional function $g(x)$. If the area under the curve is normalized so that it is equal to one, then g(X) behaves just like a probability density function, thus the general expression of the moment of order k is given by:

$$m_k = \int (x)^k g(x) dx \qquad \textbf{[eq 5-4]}$$

Since $g(k)$ represents a finite and discrete function where each bin can be assumed to have unit width and each bin element unit content, each bin content can be specified by the integer x coordinates, $x=1,2,3,...,n$. The k-th moment therefore becomes:

$$m_k = \sum_{x=1}^{n} (x)^k g(x) \qquad \textbf{[eq 5-5]}$$

Moments of inertia can be normalized in order to accommodate invariance in a number of affine transformations, for example, translation, stretching, squeezing, dilation and others.

The shape of the area that is represented by the histogram depends on the quality that each bin represents. For instance, in the LBP case that was presented earlier, each histogram bin represents a specific 8-bit pattern. The shape of the histogram is highly dependent on the order that the patterns are assigned the histogram bins. Even a slightly shifted or horizontally stretched version of a histogram can represent very different feature content. The moment is therefore only normalized over the total area in order to satisfy the conjecture of unity area under $g(x)$. It has been determined that, the second, third and fourth moments of a shape are powerful predictors of discrimination performance (Zusne 1970).

Algorithm 5.1: Pre-processing and query procedure for pre-filtering by morphological analysis of feature vectors

Pre-processing:

Step 1. The feature histograms for each image in the database are calculated

Step 2. the n-order moment for each histogram is calculated and stored

Step 3. the images are sorted with respect to their n-order moment value.

Query:

Step 1. The feature histogram and the n-order moment for the query image is calculated.

Step 2. The n-order moment of the query image is compared against the sorted database in order to find the closest match M.

Step 3. Using M as the central point, a window of N images is selected. (*i.e.* $N/2$ after M and $N/2$ before M)

Step 4. The full distance between the histogram of the query image and that of each image in N is calculated in order to retrieve the best matches.

It is evident that the accuracy of the technique is based on the discrimination ability of the n-order moment as well as the size of the window N. The latter also determines the gain in the speed of the query.

5.4 Pre-filtering type B; Boundary conditions

5.4.1 Distance Calculation as a part of CBIR

The stages of a typical CBIR system can be briefly summarized, as follows:

- Characterise the image content by extracting a number of features, for example, colour, texture and shape;
- Construct feature vectors, in which the majority of them are represented in histograms;
- Search for the closest match inside the database by: (a) calculating the distance between the feature vectors of the database images and the vector of the query image; (b) retrieving the target images with the minimum distance values.

The distance calculation among all images inside the database and the query image is an essential operational block for all content-based image retrieval. It also remains to be the least complex building block, with the highest computing processing cost in the process of searching, matching and retrieving. In order to reduce the processing cost of the distance matching process, a pre-screening mechanism is introduced to constrict the images that are compared using the distance calculation method. This is achieved by setting lower- or upper-boundaries as a safe guideline, which can be pre-calculated, to quantify the screening process and exclude the largest possible number of images that are not possibly similar to the expected targets. The processing overhead of this disqualification operation is trivial compared to the distance calculation process of the whole database.

5.4.2 Determination of boundary conditions for pre-screening

Given an image database, let q_k be the feature vector of the query image and i_k is the feature vector of a random image inside the database. The Euclidian or L_2-distance between the query feature vector q_k and image texture vector i_k is:

$$d(I,Q) = \sum_k (i_k - q_k)^2 = \sum_k i_k^2 - 2\sum_k i_k q_k + \sum_k q_k^2 = \|i_k\|^2 + \|q_k\|^2 - 2\sum_k i_k q_k \quad \textbf{[eq 5-6]}$$

where $k=1,2,\ldots, n$ refers to the individual elements of the feature vectors.

By moving $\|q_k\|$ to the left hand side of equation 5-6, it follows that:

$$d(I,Q) - \|q_k\|^2 = \|i_k\|^2 - 2\sum_k i_k q_k \quad \textbf{[eq 5-7]}$$

Since most of the feature vectors are constructed via histograms of content features and all bins of the histogram contain positive values, we have:

$$\sum_k i_k q_k \le q_{\max} \sum_k i_k \quad \textbf{[eq 5-8]}$$

where, $q_{\max}$ stands for the maximum bin inside the feature vector (histogram) of the query image Q.

This yields:

$$d(I,Q) - \|q_k\|^2 = \|i_k\|^2 - 2\sum_k i_k q_k \geq \|i_k^2\| - 2q_{\max}\sum_k i_k \quad \textbf{[eq 5-9]}$$

or:

$$d(I,Q) - \|q_k\|^2 = \|i_k\|^2 - 2\sum_k i_k q_k \leq \|i_k^2\| - 2q_{\min}\sum_k i_k \quad \textbf{[eq 5-10]}$$

$\|i_k\|^2$ represents an activity analysis for the histograms of all images inside the database, and $\sum_k i_k$ relates to its mean value.

Before the search starts, the two parameters can be pre-calculated, hence they can be regarded as two constants represented as A and B respectively. This delivers equation 5-9 and 5-10 into the following form:

$$d(I,Q) - \|q_k\|^2 \geq A - q_{\max} \times B \quad \textbf{[eq 5-11]}$$

and

$$d(I,Q) - \|q_k\|^2 \leq A - q_{\min} \times B \quad \textbf{[eq 5-12]}$$

where $A = \|i_k\|^2$ and $B = 2\sum_k i_k$.

Let any two images I_1 and I_2 with their distances to the query image Q represented as $d(I_1,Q)$ and $d(I_2,Q)$. The comparison between $d(I_1,Q)$ and $d(I_2,Q)$ is equivalent to the comparison between $d(I_1,Q) - \|q_k\|^2$ and $d(I_2,Q) - \|q_k\|^2$. As a result, a revised distance can be introduced as:

$$D(I,Q) = d(I,Q) - \|q_k\|^2 \quad \textbf{[eq 5-13]}$$

and its lower-bound and upper-bound can be represented as:

$$L = A - q_{\max} \times B \; ; \; U = A - q_{\min} \times B$$

$$\Leftrightarrow L \leq D(I,Q) \leq U$$

$$\Leftrightarrow A - q_{\max} \times B \leq D(I,Q) \leq A - q_{\min} \times B \quad \textbf{[eq 5-14]}$$

Therefore, for two revised distances $D_1(I_1,Q)$ and $D_2(I_2,Q)$ between the query image and two test images it is possible to check the following boundary conditions before calculating the full Euclidian distance:

$$\text{if } L_2 \geq U_1 \text{ then } D_2(I_2,Q) \geq D_1(I_1,Q)$$

$$\text{if } L_1 \geq U_2 \text{ then } D_1(I_1,Q) \geq D_2(I_2,Q)$$

Calculating the upper and lower boundaries only involves one multiplication and one addition, on the condition that, both A and B terms are pre-calculated. If neither of the two conditions is satisfied, direct calculation of the revised distances or their Euclidean distances would be compulsory for comparison. In this case, the computing cost could be increased instead of being reduced.

In order to design an algorithm which exploits equation 5-14 for improvement of the image comparison stage, the content-based image retrieval process described in Section 5.4.1 is reformulated as follows:

Algorithm 5.2: Outline of reformulated CBIR process

- Given a set of feature vectors, q_k for query image and $\{i_{1,k}, i_{2,k}, \ldots i_{n,k}\}$ for all n images inside the database, Euclidian distances between q_k and $n_{j,k}, j \in [1,n]$ are calculated;
- The distances are sorted in an ascending order;
- The first JJ images with the smallest distance from the query are recorded and ranked as such that the rank-1 image corresponds to the minimum distance, rank-2 image to the second smallest distance *etc.*

According to Algorithm 5.2 the problem could be greatly facilitated if the distances between the query and test images can be somehow sorted, using a process with lower processing cost that the full Euclidian distance, so that the first m images with the smallest distance are identified. Thereby, the full Euclidian distance between the query image and only those m images would be required in order to produce the exact same output as Algorithm 5.2. Based on this concept, here follows the fast searching algorithm based on the boundary conditions:

Algorithm 5.3: Pre-filtering using boundary conditions

Let:

n be the total number of images within the database.

q_k be the feature vector of the query image Q.

$i_{j,k}$, $(j=0,1,...,n)$ be the feature vector of the j^{th} image I_j inside the database

$$A = \left\| i_{j,k} \right\|^2$$

$$B = 2\sum_k i_{j,k}$$

$$L_j = A_j - q_{\max} \times B_j \; ; \; U_j = A_j - q_{\min} \times B_j ; \; j = 1,2,...,n$$

where

$k=1,2,...,\ n$ refers to the individual elements of the feature vectors.

q_{max} and q_{min} are the maximum and minimum elements of feature vector q_k respectively.

L_j and U_j are the lower and upper boundaries of the j^{th} image I_j inside the database.

Also, let L_{global} and U_{global} be the lower and upper distance boundaries which this algorithm attempts to maximize (L_{global}) and minimize (U_{global}) respectively. Only the images of which the L and U values are within the window delimited by L_{global} and U_{global} will have to be calculated using the full Euclidian distance.

Note that according to Algorithm 5.2 the first m closest matches are required to be returned by the system. Therefore one condition is for the values of L_{global} and U_{global} to "enclose" at least m images from the database. On the other hand, the closest the number of "enclosed" images is to m the greater the savings in processing power.

Step-1: Calculate L_j and U_j $(j=0,1,...,n)$ for each image in the database:

Step-2: Sort the lower boundaries in an ascending order: $L_1 < L_2 < \ldots < L_j$.

L_1 also corresponds to the global lower boundary L_{global}. This however is of little practical use by itself since it cannot be used to disqualify any of the images from the distance calculation process. The usefulness of L_{global} (effectively L_1) is that it establishes a starting point in order to begin an iterative process to determine U_{global}, as follows:

Step-3: With respect to the sorted list generated by step-2: Starting from L_1 moving to its right direction, U_1 is tested against all L_r where $r=2,3,\ldots,n$ in order to detect the point X_D where $U_1<L_r$.

To achieve further savings, this search for point X_D can be carried out via binary search within the range of $[2,n]$.

This operation reveals that according to equation 5-14 all images with Lj for $j>X_D$ have a larger distance than the image corresponding to L_1. X_D is therefore the first estimation for U_{global}.

Step-4: Repeat step-3 for all U_j where $j=1,2,\ldots,n$ against all L_r where $r=j+1,j+2,\ldots,n$.

Each time an upper-boundary is tested, four outcomes exist depending on where the corresponding X_d point is. These are summarised as follows:

a) If the X_d-point is on the right of the current estimate of U_{global}, and U_{global} already contains at least m images, the new X-point is discarded.
b) If the X_d-point is on the right of the current estimate of U_{global}, and U_{global} contains fewer than m images, the new X_d-point is regarded as the new U_{global}.
c) If the X_d-point is the same as the existing U_{global}, nothing changes;

d) If the X_d-point is on the left of the current estimate of U_{global}, there are two cases:

- if the new X_d-point contains at least m images then the new X_d-point is assigned as the new U_{global}.
- if the new X_d-point contains fewer than m images nothing changes.

The iterations stop if one of three conditions is satisfied

a) the images contained with the current window delimited by L_{global} and U_{global} are exactly m.

b) Lj reached U_{global}

Step-5: After the last iteration, all elements remaining inside the $L_{global} - U_{global}$ boundaries are uncertain, *i.e.* they cannot be further reduced using the boundary methodology. Therefore, their full distance has to be used to derive a final sorted list, which contains m elements.

The number of elements inside the final uncertainty list determines the cost saving. It must be noted that the worst case scenario is that all elements remain inside the uncertainty list. As a result, the two-pass operation with their lower-bounds and upper-bounds become an overhead for the computing cost in comparison with that of normal full distance calculation.

The effectiveness of the algorithm is therefore sensitive to the value spread of both the histograms of query image (since this defines the values of qmin and qmax) and the histograms of the images within the database (since these define the values of A and B in equation 5-14). In terms of Algorithm 5.3 these values determine the sizes of the windows delimited by low and high boundaries and thus the final uncertainty windows delimited by L_{global} and U_{global}.

A conclusion that can be derived from Algorithm 5.3 is that, starting from the image which corresponds to L1, images which correspond to L with smaller values (*i.e.* ranked higher in the sorted list) play a more significant role on the determination of U_{global} than those with higher L values. This fact could be used to enable a quick

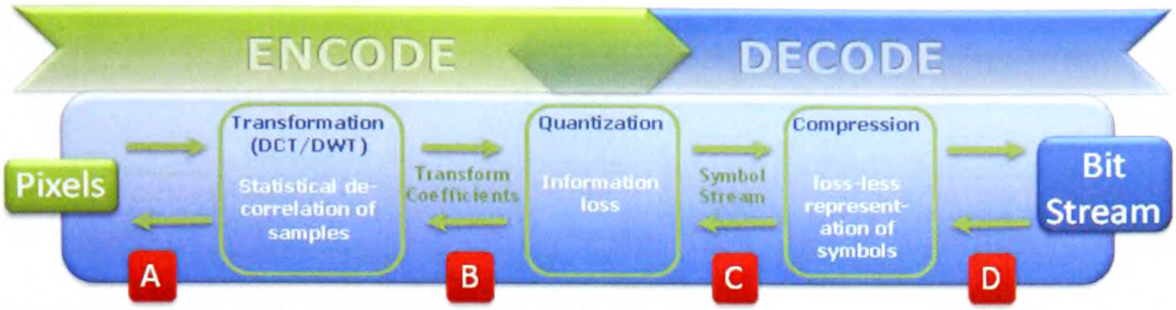


Figure 5.2: A generic model of a transform codec

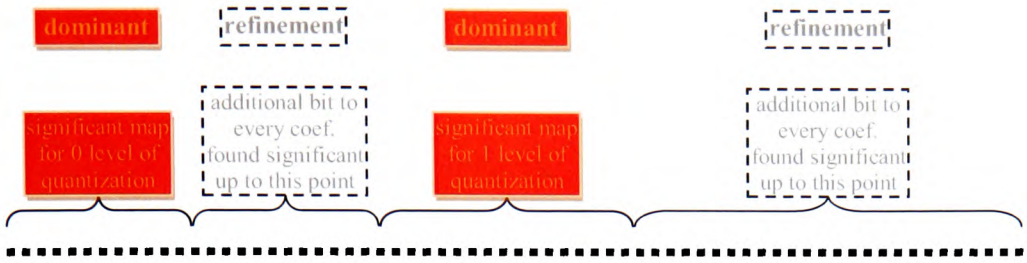
pre-processing stage which would compare the values of the first few upper boundary values with the lower boundaries which are located in the mid-high regions of the ranked list to determine if there is any potential for gain (i.e. a lot of high-L images are discarded from the first stages) and stop otherwise.

5.5 Direct feature extraction from compressed bit stream

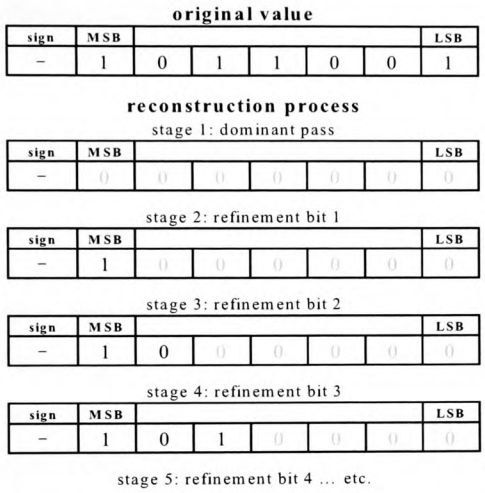
5.5.1 Introduction

In most implementations of image retrieval systems, the database contains images in compressed form for economy in storage space and network bandwidth utilisation. Therefore, in order to process the images, it is necessary to convert the compressed data into a convenient form.

The central requirement for this design is to be able to extract all necessary information without performing any decompression stage. In order to achieve that, the image database is encoded using the Set Partitioning in Hierarchical Trees (SPIHT) algorithm (Said and Pearlman 1996a). This algorithm provides one of the best performing zero-tree-based coders (Servetto, Ramchandran *et al.* 1999). The SPIHT algorithm can achieve comparable PSNRs (for the same rate or compressed file size) without using arithmetic coding at the last stage of the compression process (see "compression" stage in Figure 5.2). One key feature of SPIHT is that it is based on Significance Map Encoding or SME (see Section 3.3). SME has been proved to be a suitable basis for texture feature extraction in wavelets domain (Liang and Kuo 1999; Mandal, Idris *et al.* 1999) and therefore will serve as the starting point in the proposed system.



(a)



(b)

Figure 5.3: a) Simplified version of compressed stream using SPIHT b) First four stages of a coefficient as it is reconstructed from the compressed stream. During the dominant pass, SPIHT writes only the signs of the coefficients but in the proposed design it does not make any difference as long as the system records that the particular coefficient is significant.

5.5.2 Analysis of compression method

For the purposes of this design the focus is on two main characteristics of the SPIHT algorithm:

- Shared knowledge between the encoder and the decoder enables the recovery of the execution path, without the need to send explicit information;
- The efficiency of the algorithm allows omitting arithmetic coding without any significant loss in compression (Said and Pearlman 1996a; Servetto, Ramchandran *et al.* 1999).

During the coding process no information about the co-ordinates of the coefficients is explicitly included in the bit-stream. The bit-stream only includes information about

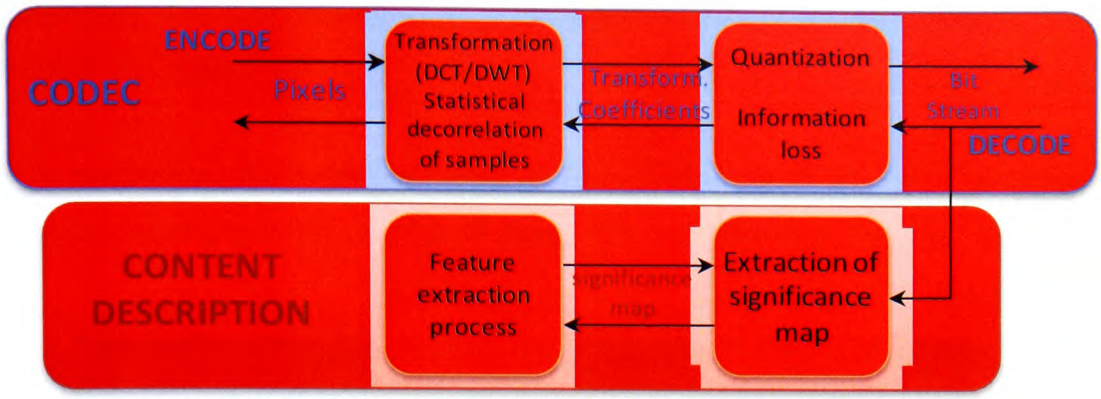


Figure 5.4: Architecture of the proposed system.

the sign and the significance of individual coefficients and coefficient groups, as well as refinement bits.

The mapping between significance/refinement bit and wavelet coefficient is deduced by a symmetric sorting algorithm, *i.e.* the encoder and the decoder have *a priori* knowledge of the order in which the bits are stored in the bit stream. Figure 5.3(a) shows a simplified model of how the bits are ordered inside the SPIHT encoded bit stream. Figure 5.3(b) shows the first four stages of a coefficient, as more refinement bits are read from the stream.

Inspection of Figure 5.3(a) reveals that, it is possible to extract information about the significance of the coefficients for a given threshold by ignoring the refinement passes. The Significance Maps provide the system with sufficient information in order to extract the texture information. Therefore, the proposed system extracts all the necessary information for content description directly from the compressed bit stream without involving any of the decompression stages. The relative reduction in processing overhead of the proposed system over existing systems becomes apparent when comparing Figures 5.2 and 5.4.

5.5.3 Analysis of the texture feature extraction method

In order to characterize the texture feature content of a compressed image, the SME principle of a *significant* coefficient is used. More specifically, the importance of each DWT subband is determined by counting the number of included significant coefficients (Liang and Kuo 1999). To this end, the feature extraction algorithm that

is presented here is an adapted version of significance-map average subband energy described in Chapter 3.

For a predetermined threshold value T the significance N_i of the i^{th} subband is measured according to the number of significant coefficients, as follows:

$$N_i(T) = \left|\{c_j \mid c_j \in S_i, c_j \geq T\}\right| \qquad \textbf{[eq 5-15]}$$

where S_i denotes the i^{th} subband and c_j the magnitude of the j^{th} coefficient.

Significance Map Encoding dictates that: after removing all coefficients that are found to be significant at one threshold level, the process is repeated using a new threshold $T_{l+1}=T_l/2$ for the remaining coefficients. In order to exploit the Significance Maps at a finer level, each threshold level is examined individually. Therefore, equation 5-15 takes the following form:

$$N'_i(T) = \left|\{c_j \mid c_j \in S_i, T_l > c_j \geq T_l\}\right| \qquad \textbf{[eq 5-16]}$$

In order to accommodate size invariance, if one level of decomposition includes m subbands, the normalized importance N_{norm} of a subband i is:

$$N_{norm,i}(T_l) = \frac{N'_i(T_l)}{\sum_{k=1}^{m} N'_k(T_l)} \qquad \textbf{[eq 5-17]}$$

Consequently we now have l texture vectors, each one corresponding to a threshold level. For m number of subbands, the texture vectors are:

$$TV(p) = \{N_{norm,1}(T_l), N_{norm,2}(T_l)..., N_{norm,m}(T_l)\} \qquad \textbf{[eq 5-18]}$$
$$, p = 1,2,..., l$$

Note that a lower value of p represents a higher threshold. The resulting feature texture descriptor (signature) of the proposed system is the concatenation of the above vectors, as follows:

$$TV = \begin{Bmatrix} N_{norm,1}(T_1), N_{norm,2}(T_1),..., N_{norm,m}(T_1), N_{norm,1}(T_2), N_{norm,2}(T_2),... \\ .., N_{norm,m}(T_2),..., N_{norm,1}(T_l), N_{norm,2}(T_l),..., N_{norm,m}(T_l) \end{Bmatrix} \qquad \textbf{[eq 5-19]}$$

Following the decompression process of SPIHT, in each dominant pass of decompression only the significant coefficients of all subbands for the particular level of decomposition are extracted. Therefore, a Significance Map of each subband for

the particular level of decomposition (*i.e.* $Nnorm_{m}(T_l)$ in eq 5-19) is already available without the need for the refinement pass. Additionally, there is no need to manually determine any of the threshold values, since these are already calculated in the compression stage. Each $TV(p)$ is constructed directly by the data collected during a SPIHT dominant pass. Equation 5-19 can thus be written for simplicity:

$$TV = \begin{Bmatrix} N_{norm,1}(1), N_{norm,2}(1),..., N_{norm,m}(1), N_{norm,1}(2), N_{norm,2}(2),... \\ ..., N_{norm,m}(2),..., N_{norm,1}(l), N_{norm,2}(l),..., N_{norm,m}(l) \end{Bmatrix} \quad \textbf{[eq 5-20]}$$

where m is the number of subbands and l the number of threshold levels.

An important effect of this approach is that, since the threshold values are determined progressively as the bit stream is read, it is possible to implement this algorithm for a successive accuracy refinement scheme. In such a scheme the images are compared and ranked for each successive threshold and the query process stops either on user request, when a specific accuracy has been achieved or when all available data have been exhausted.

Additional savings in processing time can be achieved because, the size of the refinement bit stream that lies between two dominant passes is known, therefore it can be easily skipped when data are stored in random access media. The similarity between two images is determined by the collective L_1-distance of their vector sets. The L_1-distance has been used in order to keep the complexity of the system low, since speed is of primary concern.

In SPIHT, (Said and Pearlman 1996b) elaborate on how coefficients with higher magnitude (*i.e.* being significant at higher thresholds) carry more information (in terms of their contribution in the reduction of the overall distortion), than those with lower magnitude. Therefore, higher precedence should be given to vectors representing higher thresholds (*i.e.* lower values of l). In order to take into consideration this fact, the following weight has been used when calculating the L_1-distance between two vectors:

$$w_l = \frac{1}{1+l} \quad \textbf{[eq 5-21]}$$

where, l is threshold level (as used in equation 5-20).

Another feature of DWT that is pointed out in (Liang and Kuo 1999) is that, coarser (smaller) subbands contain more important visual information than detail (larger) subbands. It follows that coarser subbands tend to contain more significant coefficients compared to the detail subbands. In order to equalize the difference in size, and therefore the smaller absolute number of significant coefficients that would appear in a coarser subband, another weighting factor is introduced. The weighting factor proposed by (Liang and Kuo 1999) and adopted here is inversely proportional to the size of a subband. The proportional size, with respect to the whole image, of each subband for the n^{th} level of decomposition is given by (see Section 3.4.1):

$$sub\ .size\ = \frac{1}{4^n} \times (image\ \_\ size\ )$$

i.e. for the first level of decomposition $n=1$ $\therefore$ each subband covers $\frac{1}{4}$ of the total image area.

Therefore, the additional weight is:

$$w_n = \frac{1}{4^n} \qquad \textbf{[eq 5-22]}$$

Concluding, the distance between the texture descriptors of two images I and Q is given by:

$$d_{L_1}(I,Q) = \sum_l w_l \left(\sum_i w_n \cdot \left| N^I_{norm\ ,i}(l) - N^Q_{norm\ ,i}(l) \right| \right) \qquad \textbf{[eq 5-23]}$$

where $N^I_{norm\ ,i}(l)$ and $N^Q_{norm\ ,i}(l)$ are the normalised subband importance vectors of images I and Q respectively at threshold level l;

w_l and w_n are the threshold (eq 5-21) and subband-size-dependent (eq 5-22) weighting factors respectively.

It is worthwhile to point out that during the calculation of $\sum_i w_n \cdot \left| N^I_{norm\ ,i}(l) - N^Q_{norm\ ,i}(l) \right|$ the value of w_n depends on the level of decomposition. Consequently, it is not constant during the calculation of the sum for a given threshold level l.

5.6 Experimental setups

5.6.1 Dataset

The retrieval experiments were performed using the in-house compiled database (Database A) of approximately 1000 images, which is presented in Chapter 3 (see Section 3.5.1.1 for more information) in order to be comparable to the respective experimental results. The image that was used as an example for querying was automatically excluded from the database. The grey scale versions of the images were obtained by transforming the original RGB values to the YUV colour space according to conversion table 3.1. The Y (Luminance) channel is then scaled and normalized to a 256 shades grey scale image.

5.6.2 Retrieval Accuracy Evaluation

The performance of the direct feature retrieval algorithm presented in Section 5.5 is quantified using average precision and the evaluation strategy which was described in Section 3.5.2.1.

5.6.3 Hardware setup

All experimental platforms were implemented on a Sun UltraSPARC® 10 running Solaris® version 9 OS. The platforms were coded in ANSI C. For pre-filtering type A and B the $LBP_{8,1}$ texture descriptor was used. The generated feature signature is a 256 bins histogram, which is normalized to unity maximum value. For all measurements the direct feature extraction process was given highest execution priority in order to restrict the effect of native system and I/O during testing.

5.7 Empirical analysis – Retrieval accuracy

Two of the presented techniques, namely, Pre-filtering type A and Direct Feature extraction, are correlated with the feature extraction technique. Pre-filtering type A reduces the actual sample pool that is used by the full retrieval process. The Direct Feature extraction technique employs a new untested feature extraction technique. In this section, the effect of the presented techniques on the accuracy of the retrieval results is investigated.

5.7.1 Pre-filtering type A – Window size vs Retrieval accuracy

The window size is an undetermined parameter of the pre-filtering type A technique. Window size is expected to affect the speed of the query. A smaller window means that fewer images need to be compared using full distance calculation. However, since pre-filtering type A is not designed to be an accurate content descriptor, very small window sizes would affect negatively the accuracy of the retrieval. Window sizes of 500, 300, 150, 100 and 50 images were used for the evaluation of the parameter. The performance of the systems is tested using the $LBP_{8,1}$ texture extraction technique as feature. The feature signature is a 256 bins histogram, which is normalized to unity maximum value. Twenty random queries were executed and the average results are shown in table 5.1.

Table 5.1: Effect of different window sizes on the retrieval performance of Pre-filtering type A method. Results expressed in % accuracy reduction over full Euclidian distance.

window size / moment	**500**	**300**	**150**	**100**	**50**
2nd order	0%	13%	47%	70%	83%
3rd order	0%	6%	28%	52%	73%
4th order	0%	6%	26%	50%	69%
combined *	0%	0%	17%	32%	61%

*** 2nd+3rd+4th moments as a 3 element vector**

Using a single moment for pre-filtering has the advantage of being able to pre-calculate and sort the values for the entire database. During a query, a binary search in the sorted list easily yields the centre point of the window to be used. In table 5.1 it is shown that the combined moment enables the use of a smaller window without considerable loss in accuracy.

It must be noted that, a combined moment is essentially a three element meta-signature that requires a distance calculation formula (*e.g.* Euclidian) for two images to be compared. In other words, using combined moment involves exactly the same procedure as normal signature calculation for a three-element signature. The following conclusions can be drawn from the results shown in table 5.1.

- Using a window of 500 images fully maintains the retrieval accuracy of the algorithm;
- For window size of 300 images, 3rd and 4th order moments have almost no effect on the accuracy;
- For window sizes of 500 and 300 images, combined moment has no effect on the accuracy and is overall the best performing alternative;
- For window sizes smaller or equal to 100, all methods greatly affect retrieval accuracy.

5.7.2 Direct feature extraction – Retrieval accuracy

The C code provided by the authors of the original SPIHT codec (Said 2006) forms the implementation basis of the direct feature extraction prototype. The code was modified in order to incorporate the feature extraction algorithm. Wrapper test routines (routines that "enclose" or "wrap" the routines under testing) were implemented in order to perform the timed (speed) experiments. The system was first tested in terms of image retrieval accuracy, in order to establish that it is not affected by the introduction of the specific compression algorithm. The retrieval accuracy results are presented in table 5.2.

Table 5.2: Retrieval accuracy of Direct feature extraction. (window of 30 matches)

	% similar results
Accuracy	59,8%
Variance	1.6

The accuracy of the method is comparable to that of the methods described in Section 3.7.3. Additionally, the results are superior to the SES algorithm which is the most comparable in terms of methodology and complexity. The low variance is indicates that the results were consistent to a high degree.

5.8 Empirical analysis – Speed improvements

5.8.1 Pre-filtering type A – Speed Improvement

Single moment calculation is essentially a one-pass search process. After establishing the position of the n^{th}-order moment of the query image inside the database, this position can be used as the centre point for the full distance calculation window. Combined moment technique requires full-distance calculation between the three-element vector signatures of the query image and the whole database, thus establishing the m-shorter distances calculation window. Note that, in both cases the constant term omission that is described in Section 5.3.1 accounts for approximately 18%-19% of the speed improvement over full L_2-distance.

Table 5.3: Gain in total time required for a query. Results expressed as % improvement over full Euclidian distance without any pre-filtering.

window size / moment	**500**	**300**	**150**	**100**	**50**
single order	55%	74%	81%	88%	92%
combined *	54%	73%	79%	86%	90%

*** 2^{nd}+3^{rd}+4^{th} moments as a 3 element vector**

The results that are shown in table 5.3 show that the improvement in time is approximately proportional to the reduction of the database sample that is ultimately used for full distance calculation. It is worth noting that, although the combined moment requires full distance calculation for the whole database, the processing time cost is not much higher than single-moment. It must be noted that the file/memory I/O involved in the process accounts for a significant overhead. The following concluding remarks are made:

- Combined moment is slower than single moment calculation;
- The difference between single moment and combined moment is small (approximately 1%-2%).

5.8.2 Pre-filtering type B - Speed Improvement

In order to perform the speed tests for pre-filtering type B, a threshold of 30 images for the entirety of the clusters was used. To formulate this, the following to rules were used:

1. If the first cluster contains ≥ 30 images, then all remaining clusters are discarded.
2. If the first cluster contains < 30 images, then more clusters are added to the full distance calculation stage, until the 30 images threshold is exceeded.

Note that, if cluster 1 has < 30 images and the sum of cluster 1 and cluster 2 exceeds 30 images, then all images contained in both clusters are processed through full distance calculation.

Table 5.4: Timed experiments for Pre-filtering type B.

query	Full Euclidian (msec)	Pre-filtering type B (msec)	Difference (msec)	Saving (%)
1	180	55	125	69%
2	177	68	109	62%
3	183	105	78	43%
4	182	178	4	2%
5	180	215	-35	-19%
6	172	95	77	45%
7	179	73	106	59%
8	178	54	124	70%
9	180	58	122	68%
10	184	64	120	65%
AVERAGE	179,5	96,5	83	46,30%
STD DEV	3,41	55,97	55,54	30,88%

Table 5.4 contains indicative results from ten experiments. Overall the reported results indicate improvement of the query speed. Two notable exceptions are queries 4 (very small improvement) and 5 (overhead instead of improvement). Observation of the results yields the following concluding comments:

- Pre-filtering type B can offer considerable speed gain, up to approximately 70% over full Euclidian distance.
- The high standard deviation value indicates that, pre-filtering type B method is very sensitive to the statistical properties of the feature histogram.
- If the uncertainty list defined by the generated clusters only excludes a small part of the image database, pre-filtering type B can result in an overhead instead of a gain to the retrieval process.

5.8.3 Direct feature extraction – Speed Improvement

The gain in retrieval speed was tested on the direct feature extraction prototype by comparison of: a) the time required in order to complete a query using the direct feature extraction technique, and b) the time required to complete a query by first building the complete wavelet decomposition tree from the compressed data stream, followed by the feature extraction stage. To provide meaningful results, no pre-processing was performed for either technique. Therefore, the whole database had to be processed in each query.

Full decomposition retrieval involves two steps: a) transfer of each image from database storage to RAM and b) create feature signature. The respective steps required in Direct Feature retrieval are: a) transfer only 30% of the most significant coefficients of each image from database storage to RAM and b) dynamically create feature signature. Distance calculation and results sorting stages are equivalent in both systems.

Direct feature extraction is directly affected by random access data I/O overhead because it involves a lot of data "leaps". Depending on the file system used, fragmented files can also heavily affect full decomposition retrieval. In the implemented prototype, the image database was stored in a continuous segment of the hard drive to reduce the effect of random I/O access seek time.

The results are expressed as the ratio of *[time required to complete direct feature extraction query] / [time required to complete full decomposition query]* for the same query image. Indicative results of 10 queries are shown in table 5.5.

Table 5.5: Gain in the speed of a query. Results expressed as time required for direct feature extraction query as a percentage (%) of the time required for the full decomposition query.

experiment	1	2	3	4	5	6	7	8	9	10
duration %	52%	55%	62%	60%	60%	61%	56%	53%	50%	58%

As aforementioned, the dependence of the system on the actual distribution of the significance coefficients in the data stream can be held accountable for the variation in the resulted performance. If many significant coefficients are gathered in the first blocks of data read by the data stream, the >30% significance condition is satisfied faster and so is the overall duration of the algorithm execution.

Overall, factors such as, system I/O, the non-determinism of the operating system and the implementation particularities of the programming language used, affect the actual performance of the system and add to the entropy of the experimental results. Therefore, the results give indicative, albeit solid, indications, rather than an absolute assessment of the performance. The reported results suggest that the proposed algorithm can considerably improve the speed of a query, whereas leads for further analysis are given, in order to accurately quantify the exact gain that can be achieved.

5.9 Conclusions

In this chapter, a number of methods were presented, which address two of the original research aims of this research namely: a) to suggest methods for improving the speed of CBIR/CBIC systems, and b) to extract feature directly from the compressed bit-stream. The aims are building upon the conclusion drawn from the exposure to the research literature that a) there is an abundance of space for research in CBIR speed improvement and, b) there are unresolved issues with existing approaches for direct feature extraction from compressed images, particularly the requirement for some level of pre-processing from the decoder. Part of the work presented in this Chapter on pre-filtering type A and direct feature extraction from compressed images has been presented in national and international refereed conferences (Voulgaris and Jiang 2001a; Voulgaris and Jiang 2001b; Voulgaris and Jiang 2002b; Voulgaris, Ware et al. 2002)

The first two techniques build upon the principle of dividing the signature pool in order to introduce different ways for trisection of the feature histogram space. The first technique is treating the histogram signatures as solid shapes and performs a moments-based morphological analysis in order to disqualify part of the database from the full distance calculation process. The morphological comparison has very low processing cost, compared to L_2-distance calculation.

The second technique exploits the way the L_2-distance formula is used in establishing similarity in an image database in order to establish two opposing boundaries which are enclosing a sub-set of the entire signature database. The signatures that are enclosed by the boundaries are then compared using L_2-distance.

The problem of compression and image retrieval is addressed jointly by the third technique, named Direct Feature extraction, which enhances the current state-of-the-art by extracting features directly from the bit-stream. The method exploits the characteristics of the SPIHT wavelet-based compression algorithm in order to overcome many of the shortcomings of existing on-the-fly feature extraction techniques. In particular, the technique extracts texture feature directly from the bit-stream without any processing required from the decoder and has the ability to offer progressive accuracy refinement. An important differentiator of these advantages is that existing techniques require modification of the codec or introduction of extra symbols in order to achieve them. The extracted information is used to dynamically create a texture signature. The employed feature extraction algorithm builds upon the existing method of Significance-map-based Energy Signature (SES) which is enhanced in order to deliver better accuracy and the ability for progressive refinement of accuracy. However, the Direct Feature extraction can be applied to any significance map-based feature extraction technique.

Finally, based on the particularities of the way L_2-distance is applied in CBIR, a number of observations led to a computationally simplified and faster technique for similarity measurement.

The speed optimization techniques that were presented in this work address issues associated with the pre-calculation and the comparison (distance calculation) of

feature signatures. The three techniques offer a diversity of approaches, namely, a method for pre-filtering of the signature database, a method for faster calculation of the L_2-distance and a method for feature extraction without decompression of the stored images.

EMPIRICAL ANALYSIS

Two versions of the Pre-filtering Type A method are presented using either single or multiple moments for morphological characterization of the signature histogram. Empirical findings showed that narrowing the full distance calculation sample set to half of the original fully maintains the retrieval accuracy of the feature extraction and indexing algorithm. This corresponds approximately to 50% gain in processing time. Narrowing the full distance calculation sample set to less than half the size of the original available pool affects negatively the accuracy of the retrieval. The combined moment variation is the only exception which may allow reduction down to 1/3 of the database and proportional savings in speed. For a window size of approximately 1/3, single moment methods are faster by approximately 1%-2% over combined moments with approximately 6%-13% loss in accuracy. This also gives an indication of the overhead dependency of the timed experiments on the system I/O processes. Overall, the combined moment variation offers the best ratio of accuracy drop over retrieval speed gain.

Significant savings in time, up to 70% faster than full L_2-distance, are recorded in the empirical analysis of Pre-filtering Type B. This technique is completely uncorrelated to the retrieval accuracy of the system. The performance of the algorithm fluctuates considerably depending on the statistics of the feature histogram. Under certain conditions, there is the possibility that the boundaries may enclose a very large proportion or even the entire sample set. In those cases, the process may potentially delay, instead of speed up, the retrieval process.

The achieved gain in speed of the Direct Feature Extraction method over full decompression and feature extraction reached approximately 60% and remained fairly consistent throughout the experiments. Any fluctuations can be attributed to the non-deterministic system I/O processes. The retrieval accuracy of the feature extraction and indexing algorithm is comparable to existing CBIR examples, such as

those presented in Chapter 3, and superior to the directly comparable SES descriptor.

All three approaches are promising starting points for further work. The results indicate that further analysis of the algorithms might lead to quantification of the performance in more detail, for example, in terms of number of operations. Detail study of the parameters that cause any fluctuation in the performance of the three methods might also lead to further improvement in retrieval accuracy, tolerance and speed.

Chapter 7: References

Ahmad, I., S. Kiranyaz and M. Gabbouj (2005). An Efficient Image Retrieval Scheme on Java Enabled Mobile Devices. Multimedia Signal Processing, 2005 IEEE 7th Workshop on Shanghai.

Alt, F. (1962). "Digital Pattern Recognition by Moments." Journal of the ACM **9**: 240-258.

AltaVista. (2008). "AltaVista image search." Last access date 01/06/08, from http://www.altavista.com/image/default.

Amsaleg, L. and P. Gros (2001). "Content-based Retrieval Using Local Descriptors: Problems and Issues from a Database Perspective." Pattern Analysis & Applications **4**(2): 108-124.

Annabelle, C., P. Patrick and rez (1999). Unsupervised Image Classification with a Hierarchical EM Algorithm. Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2, IEEE Computer Society.

Antonini, M., M. Barlaud, P. Mathieu and I. Daubechies (1992). "Image coding using wavelet transform." IEEE Trans. on Image Processing **1**(2): 205-220.

Au, K. M., N. F. Law and W. C. Siu (2007). "Unified feature analysis in JPEG and JPEG 2000-compressed domains." Pattern Recognition **40**(7): 2049-2062.

Autonomy. (2008). "Corporate web-site." Last access date 01/06/08, from http://www.autonomy.com.

Avci, E. (2008). "Comparison of wavelet families for texture classification by using wavelet packet entropy adaptive network based fuzzy inference system." Applied Soft Computing **8**(1): 225-231.

Bach, J. R., C. Fuller, A. Gupta, A. Hampapur, B. Horowirz, R. Humphrey, R. Jain and C.-F. Shu (1996). The Virage image search engine: an open framework for image management. SPIE Storage and Retrieval for Image and Video databases IV, San Jose CA, U.S.A., SPIE.

Barcelos, C. A. Z., M. J. R. Ferreira and M. L. Rodrigues (2007). "Retrieval of textured images through the use of quantization and modal analysis." Pattern Recognition **40**(4): 1195-1206.

Bashar, M. K., T. Matsumoto and N. Ohnishi (2003). "Wavelet transform-based locally orderless images for texture segmentation." Pattern Recognition Letters **24**(15): 2633-2650.

Bashir, F., S. Khanvilkar, A. Khokhar and D. Schonfeld (2003). Content Based Indexing and Retrieval. Multimedia Systems.

Benchathlon. (2008). "The Benchathlon Network : CBIR Benchmarking." Last access date 01/06/08, from http://www.benchathlon.net.

Berman, A. P. and L. G. Shapiro (1999). "A flexible image database system for content-based retrieval." Computer Vision and Image Understanding, Academic Press **75**(1/2): 175-195.

Birney, K. A. and T. R. Fischer (1995). "On the modeling of DCT and subband image data for compression." IEEE Trans. on Image Processing **4**(2): 186-193.

Bishop, C. M. (1995). Neural networks for pattern recognition. Oxford; New York, Clarendon Press ; Oxford University Press.

Bishop, C. M. (2005). Neural Networks for Pattern Recognition, Oxford University Press.

Biswas, M., M. Frater and J. Arnold (2007). Multiple Description Video Codingwith 3D-Spiht Employing a New Tree Structure. IEEE International Conference on Image Processing (ICIP 2007). San Antonio, TX, IEEE. **3:** 389-392.

Bo, S. and I. K. Sethi (1996). Direct feature extraction from compressed images. Storage and Retrieval for Still Image and Video Databases IV. San Jose, CA.

Bovik, A. C., M. Clark and W. S. Geisler (1990). "Multichannel texture analysis using localized spatial filters." IEEE Transactions on Pattern Analysis and Machine Intelligence **12**(1): 55-73.

Brady, M. and Z.-Y. Xie, Eds. (1996). Feature Selection for Texture Segmentation. Advances in Image Understanding, IEEE Computer Society Press.

Brodatz, P. (1966). Textures: A photographic album for artists and designers. New York, U.S.A, Dover.

Brown, L. and L. Gruenwald (1998). "Tree-Based Indexes for Image Data." Journal of Visual Communication and Image Representation **9**: 300-313.

Burges, C. J. C. (1998). "Tutorial on Support Vector Machines for Pattern Recognition." Data Mining and Knowledge Discovery **2**(2): 121-167.

Bytemobile. (2008). "Corporate web-site." Last access date 01/06/08, from http://bytemobile.com.

Caicedo, J., F. González, E. Triana and E. Romero (2007). Design of a Medical Image Database with Content-Based Retrieval Capabilities. Advances in Image and Video Technology**:** 919-931.

Campbell, F. W. and J. G. Robson (1968). "Application of Fourier Analysis to the Visibility of Gratings." Journal of Physiology **197**: 551-566.

Cariou, C. and K. Chehdi (2008). "Unsupervised texture segmentation/classification using 2-D autoregressive modeling and the stochastic expectation-maximization algorithm." Pattern Recognition Letters **29**(7): 905-917.

Carson, C., S. Belongie, H. Greenspan and J. Malik (Aug 2002). "Blobworld: Image Segmentation using Expectation Maximization and its Application to Image Querying." IEEE PAMI **24**(8).

Castellano, G., L. Bonilha, L. M. Li and F. Cendes (2004). "Texture analysis of medical images." Clinical Radiology **59**(12): 1061-1069.

Chang, S. and L. Carin (2006). "A modified SPIHT algorithm for image coding with a joint MSE and classification distortion measure." Image Processing, IEEE Transactions on **15**(3): 713-725.

Chang, S. F. (1995). Compressed Domain Techniques for Image/Video Indexing and Manipulation. IEEE International Conference on Image Processing.

Chang, T. and C.-C. J. Kuo (1993). "Texture Analysis and Classification with Tree-Structured Wavelet Transform." IEEE Transactions on Image Processing **2**(4): 429-441.

Chang, W. and J. Wang (1999). Metadata for multi-level content-based retrieval. 3rd IEEE Meta-Data Conference.

Chapelle, O., P. Haffner and V. N. Vapnik (1999). "Support vector machines for histogram-based image classification." Neural Networks, IEEE Transactions on **10**(5): 1055-1064.

Chatzichristofis, S. A. and Y. S. Boutalis. (2008a). "Anaktisi web-site." Last access date 01/06/08, from http://orpheus.ee.duth.gr/anaktisi/#.

Chatzichristofis, S. A. and Y. S. Boutalis (2008b). CEDD: Color and edge directivity descriptor - a compact descriptor for image indexing and retrieval. 6th International Conference in advanced reserach on Computer Vision Systems ICVS 2008, Santorini, Greece, Elsevier.

Chaudhuri, B. B. and N. Sarkar (1995). "Texture segmentation using fractal dimension." IEEE Transactions on Pattern Analysis and Machine Intelligence **17**(1): 72-7.

Chen, W.-Y. (2008). "Color image steganography scheme using DFT, SPIHT codec, and modified differential phase-shift keying techniques." Applied Mathematics and Computation **196**(1): 40-54.

Chen, Y.-Y. (2007). "Medical images compression for remote diagnosis using modified SPIHT data organization and fidelity enhancement filter." Int. J. Imaging Syst. Technol. **17**(2): 49-61.

Chen, Y. and E. Dougherty (1994). "Grey-Scale Morphological Granulometric Texture Classification." Optical Engineering **33**(8): 2713-2722.

Cheng, K. O., N. F. Law and W. C. Siu (2007). "Multiscale directional filter bank with applications to structured and random texture retrieval." Pattern Recognition **40**(4): 1182-1194.

Cheng, P.-C., B.-C. Chien, H.-R. Ke and W.-P. Yang (2008). "A two-level relevance feedback mechanism for image retrieval." Expert Systems with Applications **34**(3): 2193-2200.

Chevallet, J. P., J. H. Lim and M. K. Leong (2007). "Object identification and retrieval from efficient image matching. Snap2Tell with the STOIC dataset." Information Processing and Management **43**(2): 515-530.

Chitre, Y. and A. P. Dhawan (1999). "M-band Wavelet Discrimination of Natural Texture." Pattern Recognition **32**: 773-789.

Chong, C.-W. and P. Raveendran (2003). "Translation Invariants of Zernike moments." Pattern Recognition **32**: 1765-1773.

Clough, P., H. Mueller and M. Sanderson (2005). The CLEF Cross Language Image Retrieval Track (ImageCLEF). Lecture Notes in Computer Science (LNCS). Heidelberg, Germany, Springer.

Colak, T. and R. Qahwaji (2007). "Automated Prediction of Solar Flares Using Neural Networks and Sunspots Associations." Advances in Soft Computing **39**: 316-324.

Convera. (2008). "Corporate web-site." Last access date 01/06/08, from http://www.convera.com.

Corel. (2008). "COREL Stock Photo Libraries." Last access date 02/06/08, from http://store.corel.com/webapp/wcs/stores/servlet/ProductDisplay?productId=89703&catalogId=10805&storeId=10302&langId=-11.

Corral, A., A. D'Ermiliis, Y. Manolopoulos and M. Vassilakopoulos (2005). VA-Files vs. R*-Trees in Distance Join Queries. Advances in Databases and Information Systems: 153-166.

Datta, R., D. Joshi, J. Li and J. Z. Wang (2008). "Image Retrieval: Ideas, Influences, and Trends of the New Age." ACM Transactions on Computing Surveys **40**(2).

Daubechies, I. (1992). Ten Lectures on Wavelets. Philadelphia, SIAM.

Daugman, J. (1985). "Uncertainty Relation for Resolution in Space, Spatial Frequency and Orientation Optimised by Two-Dimensional Visual Cortical Filters." Journal of the Optical Society of America **2**: 1160-1169.

delicious. (2008). "del.icio.us Social Bookmarking." Last access date 02/06/08, from http://delicious.com/ & http://del.icio.us.

Descampe, A., P. Vandergheynst, C. De Vleeschouwer and B. Macq (2006). Coarse-to-fine textures retrieval in the JPEG 2000 compressed domain for fast browsing of large image databases. Proc. MRCS 2006.

Deselaers, T., D. Keysers and H. Ney (2004). Features for Image Retrieval: A Quantitative Comparison. Pattern Recognition: 228-236.

Deselaers, T., D. Keysers and H. Ney (2008). "Features for image retrieval: an experimental comparison." Information Retrieval **11**(2): 77-107.

Dettori, L. and L. Semler (2007). "A comparison of wavelet, ridgelet, and curvelet-based texture classification algorithms in computed tomography." Computers in Biology and Medicine **37**(4): 486-498.

Deuel, R. (2004). "Multimedia search: Ready or Not?" IEEE Distributed Systems Online **5**(7): 1-7.

Devijver, P. A. and J. Kittler (1982). Pattern recognition: a statistical approach. Englewood Cliffs, New Jersey, Prentice Hall.

Do, M. N. and M. Vetterli (2002a). "Rotation invariant texture characterization and retrieval using steerable wavelet-domain hidden Markov models." Multimedia, IEEE Transactions on **4**(4): 517-527.

Do, M. N. and M. Vetterli (2002b). "Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance." Image Processing, IEEE Transactions on **11**(2): 146-158.

Duda, R. O. and P. E. Hart (1973). Pattern classification and scene analysis, J. Wiley and Sons.

Eakins, J. and M. Graham (1999). Content-based Image Retrieval, University of Northumbria at Newcastle.

Efron, B. and G. Gong (1983). "A Leisurely Look at the Bootstrap, the Jackknife, and Cross-Validation." The American Statistician **37**(1): 36-48.

Fahmy, G., J. Black, Jr. and S. Panchanathan (2006). "Texture characterization for joint compression and classification based on human perception in the wavelet domain." Image Processing, IEEE Transactions on **15**(6): 1389-1396.

FastSearch. (2008). "Corporate web-site." Last access date 01/06/08, from http://www.fastsearch.com.

Feng, H., R. Shi and T.-S. Chua (2004). A bootstrapping framework for annotating and retrieving WWW images. Proceedings of the ACM International Conference on Multimedia.

Flickner, M., H. Sawhney, W. Niblack, J. Ashley, H. Qian, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele and P. Yanker (1995). "Query by image and video content: the QBIC system." Computer **28**(9): 23-32.

Flickr. (2008). "Flickr Photo Sharing." Last access date 02/06/08, from http://www.flickr.com/.

Fountain, S. R. and T. N. Tan (1997). Extraction of noise robust rotation invariant texture features via multichannel filtering. International Conference on Image Processing (ICIP'97).

Fowler, J. E. (2008). "QccPack graphics library." Last access date 01/06/08, from http://sourceforge.net/projects/qccpack.

Freund, Y. and R. Schapire (1997). "A decision-theoretic generalization of on-line learning and an application to boosting." J. Comput. Syst. Sci. **55**: 119-139.

Fukunaga, K. (1990). Introduction to Statistical Pattern Recognition. London, Academic Press.

Gang, Z. and M. Zong-Min (2007). Texture feature extraction and description using gabor wavelet in content-based medical image retrieval. Wavelet Analysis and Pattern Recognition, 2007. ICWAPR '07. International Conference on.

Ghanbari, M. (2003). Standard Codecs: Image Compression to Advanced Video Coding. London, IET.

Ghouti, L., A. Bouridane and D. Crookes (2006). "Edge-directed invariant shoeprint image retrieval." IET Conference Publications **2006**(CP522): 58-61.

Giorgio, G. (2007). A nearest-neighbor approach to relevance feedback in content based image retrieval. Proceedings of the 6th ACM international conference on Image and video retrieval. Amsterdam, The Netherlands, ACM.

Goldberger, J., H. Greenspan and S. Gordon (2003). An efficient similarity measure based on approximations of KL-divergence between two Gaussian mixtures. International Conference on Computer Vision (ICCV).

Gong, G. (1986). "Cross-validation, the jackknife and the bootstrap excess error estimation in forward regression logistic regression." Journal of American Statistical Association **81**(393): 108-113.

Google. (2008a). "Google Image Search." Last access date 01/02/08, from http://images.google.com/.

Google. (2008b). "Google Scholar search engine." Last access date 01/06/08, from http://scholar.google.com.

Google. (2008c). "Google www search engine." Last access date 02/06/08, from http://www.google.com.

Greene, K. (2007, 16/1/2007). "Novel Chip Architecture Could Extend Moore's Law." MIT Technology Review Last access date 01/06/08, from http://www.technologyreview.com/Infotech/18063/.

Gupta, L., S. Lekshmi, J. Majumdar and S. Das (2006). "Study of the performance of different texture features for classification of SAR images." IET Conference Publications **2006**(CP522): 315-320.

Hadjidemetriou, E., M. D. Grossberg and S. K. Nayar (2004). "Multiresolution histograms and their use for recognition." IEEE Transactions on Pattern Analysis and Machine Intelligence **26**(7): 831-847.

Han, Y. and P. Shi (2007). "An adaptive level-selecting wavelet transform for texture defect detection." Image and Vision Computing **25**(8): 1239-1248.

Haralick, R. (1979). "Statistical and Structural Approaches to Texture." Proc. IEEE **65**(5): 786-804.

Haralick, R. and L. G. Shapiro (1992). Computer and Robot Vision, vol. I. Reading, MA, Addison-Wesley.

Hastie, T., R. Tibshirani and J. Friedman (2001). The Elenments of Statistical Learning: Data Mining, Inference, and Prediction. New York, Springer Verlag.

He, X., J. Li, W. Jia, Q. Wu and T. Hintz (2007). Local Binary Patterns on Hexagonal Image Structure. Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on. Aizu-Wakamatsu, Fukushima, Japan, IEEE.

Heikkila, M. and M. Pietikainen (2006). "A texture-based method for modeling the background and detecting moving objects." Pattern Analysis and Machine Intelligence, IEEE Transactions on **28**(4): 657-662.

Hiremath, P. S. and S. Shivashankar (2008). "Wavelet based co-occurrence histogram features for texture classification with an application to script identification in a document image." Pattern Recognition Letters **29**(9): 1182-1189.

Houchin, J. S. and D. W. Singer (2002). "File format technology in JPEG 2000 enables flexible use of still and motion sequences." Signal Processing: Image Communication **17**(1): 131-144.

Huang, D., Y. Wang and Y. Wang (2007). A Robust Method for Near Infrared Face Recognition Based on Extended Local Binary Pattern. Advances in Visual Computing: 437-446.

Huang, P. W., S. K. Dai and P. L. Lin (2006). "Texture image retrieval and image segmentation using composite sub-band gradient vectors." Journal of Visual Communication and Image Representation **17**(5): 947-957.

Huijmans, D. P. and N. Sebe (2005). "How to complete performance graphs in content-based image retrieval: Add generality and normalize scope." IEEE Trans. Pattern Analysis and Machine Intelligence **27**(2): 245-251.

IBM. (2008). "IBM DB2 product web-page." Last access date 01/06/08, from http://domino.watson.ibm.com/comm/pr.nsf/pages/rsc.qbic.html.

IllusionWorks. (2008). "Perceptual Ambiguity." Last access date 02/06/08, from http://www.psychologie.tu-dresden.de/i1/kaw/diverses%20Material/www.illusionworks.com/html/perceptual_ambiguity.html.

ITU-R (2007). Recommendation BT.601-5 "Studio Encoding Parameters of Digital Television for Standard 4:3 and Wide-Screen 16:9 Aspect Ratios", International Telecommunication Union.

Jacobs, C. E., A. Finkelstein and D. H. Salesin (1995). Fast multiresolution image querying. Proceedings of SIGGRAPH '95. Los Angeles, CA.

Jafari-Khouzani, K. and H. Soltanian-Zadeh (2005a). "Radon transform orientation estimation for rotation invariant texture analysis." Pattern Analysis and Machine Intelligence, IEEE Transactions on **27**(6): 1004-1008.

Jafari-Khouzani, K. and H. Soltanian-Zadeh (2005b). "Rotation-invariant multiresolution texture analysis using Radon and wavelet transforms." Image Processing, IEEE Transactions on **14**(6): 783-795.

Jain, A. K. (1989). Fundamentals of Digital Image Processing. Englwood Cliffs, New Jersy, Prentice Hall.

Jiang, J., A. Armstrong and G. C. Feng (2002). "Direct content access and extraction from JPEG compressed images." Pattern Recognition **35**(11): 2511-19.

Jiang, J., A. Armstrong and G. C. Feng (2004). "Web-based image indexing and retrieval in JPEG compressed domain." Multimedia Systems **9**(5): 424-32.

Jiang, J., M. G. Liu and C. H. Hou (2003). "Texture-based image indexing in the process of lossless data compression." Vision, Image and Signal Processing, IEE Proceedings **150**(3): 198-204.

Julesz, B. (1975). "Experiments in the Visual Perception of Texture." Scientific American **232**(4): 34-43.

Kaplan, L. M. and C. C. Kuo (1995). "Texture roughness analysis and synthesis via extended self-similar (ESS) model." IEEE Transactions on Pattern Analysis and Machine Intelligence **17**(11): 1043-56.

Kashyap, R. L. and A. Khotanzad (1986). "A Model-based Method for Rotation Invariant Texture Characterization." Pattern Analysis and Machine Intelligence, IEEE Transactions on **8**: 472-481.

Keister, L. H., Ed. (1994). User types and queries: impact on image access systems. Challenges in Indexing electronic text and images, ASIS.

Kevin L. Priddy and P. E. Keller (2005). Artificial Neural Networks, SPIE Press.

Kherfi, M. L., D. Ziou and A. Bernardi (2003). "Combining positive and negative examples in relevance feedback for content-based image retrieval." Journal of Visual Communication and Image Representation **14**(4): 428-457.

Kherfi, M. L., D. Ziou and A. Bernardi (2004). "Image Retrieval from the World Wide Web: Issues, Techniques, and Systems." ACM Comput. Surv. **36**(1): 35-67.

Kokare, M., P. K. Biswas and B. N. Chatterji (2005). "Texture image retrieval using new rotated complex wavelet filters." Systems, Man and Cybernetics, Part B, IEEE Transactions on **35**(6): 1168-1178.

Kokare, M., P. K. Biswas and B. N. Chatterji (2007). "Texture image retrieval using rotated wavelet filters." Pattern Recognition Letters **28**(10): 1240-1249.

Kurita, T. and T. Kato (1993). Learning of personal visual impression for image database systems. Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on, IEEE.

L1id. (2008). "L-1 Identity Solutions corporate web-site." Last access date 01/06/08, from http://www.l1id.com.

Lee, D. T. (2005). "JPEG 2000: Retrospective and New Developments." Proceedings of the IEEE **93**(1): 32-41.

Lerski, R. A., K. Straughan, L. R. Schad, D. Boyce, S. Bluml and I. Zuna (1993). "Tissue characterization by magnetic resonance spectroscopy and imaging: results of a concerted research project of the European Economic Community. VIII. MR image texture analysis-an approach to tissue characterization." Magnetic Resonance Imaging **11**(6): 873-87.

Levine, M. (1985). Vision in Man and Machine, McGraw-Hill.

Lew, M. S., N. Sebe, C. Djeraba and R. Jain (2006). "Content-based multimedia information retrieval: State of the art and challenges." ACM Trans. Multimedia Comput. Commun. Appl. **2**(1): 1-19.

Li, J. and C.-C. Jay Kuo (1999). "Image Compression with a Hybrid Wavelet-Fractal Coder." IEEE Transactions on Image Processing **8**(6).

Lian, C. J., K. F. Chen, H. H. Chen and L. G. Chen (2003). Analysis and Architecture Design of Block-Coding Engine for EBCOT in JPEG 2000. **13:** 219.

Liang, K.-C. and C.-C. J. Kuo (1999). "WaveGuide: A Joint Wavelet-Based Image Representation and Description System." IEEE Transactions on Image Processing **8**(11): 1619-1629.

Liapis, S. and G. Tziritas (2004). "Color and texture image retrieval using chromaticity histograms and wavelet frames." Multimedia, IEEE Transactions on **6**(5): 676-686.

Lillo, A. D., G. Motta and J. A. Storer (2008). "Multiresolution Rotation-Invariant Texture Classification Using Feature Extraction in the Frequency Domain and Vector Quantization." Data Compression Conference, 2008. DCC 2008: 452-461.

Lin, H.-D. (2007). "Automated visual inspection of ripple defects using wavelet characteristic based multivariate statistical approach." Image and Vision Computing **25**(11): 1785-1801.

Lin, T. W. (1997). "Compressed quadtree representations for storing similar images." Image and Vision Computing **15**: 833-843.

Liu, Y., D. Zhang, G. Lu and W. Y. Ma (2007). "A survey of content-based image retrieval with high-level semantics." Pattern Recognition **40**(1): 262-282.

LTU. (2008). "LookThatUp Corporate web-site." Last access date 01/06/08, from http://www.ltutech.com/en/.

Lu, C. (1997). "Unsupervised Texture Segmentation via Wavelet Transform." Pattern Recognition **30**(5): 729-742.

Lycos. (2008). "Lycos image search." Last access date 01/06/08, from http://search.lycos.com/?tab=multi&cat=images&.

Mallat, S. (1989). "Multifrequency Channel Decomposition of Images and Wavelet Models." IEEE Trans. on Acoustic, Speech and Signal Processing **37**(12): 2091-2110.

Mandal, M. K., T. Aboulnasr and S. Panchanathan (1996). Image indexing using moments and wavelets. 1996 International Conference on Consumer Electronics. Rosemont, IL.

Mandal, M. K., F. Idris and S. Panchanathan (1999). "A Critical Evaluation of Image and Video Indexing Techniques in the Compressed Domain." Image and Vision Computing Journal **17**(7): 513-529.

Mandal, M. K. and C. Liu (2004). "Efficient image indexing techniques in the JPEG2000 domain." Journal of Electronic Imaging **13**(1): 182-190.

Mandal, M. K., S. Panchanathan and T. Aboulnasr (1997). Image indexing using translation and scale-invariant moments and wavelets. Storage and Retrieval for Image and Video Databases V. San Jose, CA.

Mandal, M. K., S. Panchanathan and T. Aboulnasr (1998). "Illumination Invariant Image Indexing using moments and wavelets." Journal of Electronic Imaging **7**(2): 282-293.

Manjunath, B. S. and W. Y. Ma (1996). "Texture features for browsing and retrieval of image data." IEEE Transactions on Pattern Analysis and Machine Intelligence **18**(8): 837-842.

Markoff, J. (2007, 27/1/2007). "Intel Says Chips Will Run Faster, Using Less Power." The New York Times Last access date 01/06/08, from http://www.nytimes.com/2007/01/27/technology/27chip.html?_r=1&em&ex=1170046800&en=59a4d10473c4a8c8&ei=5087%0A&oref=slogin#.

Martin, K., R. Lukac and K. N. Plataniotis (2006). "SPIHT-Based Coding of the Shape and Texture of Arbitrarily Shaped Visual Objects." Circuits and Systems for Video Technology, IEEE Transactions on **16**(10): 1196-1208.

Materka, A. and M. Strzelecki (1998). Texture Analysis Methods - A Review. Brussels, Technical University of Lodz, Institute of Electronics.

Megalooikonomou, V., J. Zhang, D. Kontos and P. R. Bakic (2007). Analysis of texture patterns in medical images with an application to breast imaging. Medical Imaging 2007: Computer-Aided Diagnosis, San Diego, CA, USA, SPIE.

Melendez, J., D. Puig and M. Garcia (2007). Comparative Evaluation of Classical Methods, Optimized Gabor Filters and LBP for Texture Feature Selection and Classification. Computer Analysis of Images and Patterns**:** 912-920.

Mezaris, V., I. kompatsiaris and M. G. Strintzis (2003). "An ontology approach to object-based image retrieval." Proceedings of the ICIP, vol. II: 511-514.

Microsoft. (2008). "MSN Live Search." Last access date 01/06/08, from http://search.msn.com/images.

Miguel Angel, G., Dom and P. nec (2007). "Supervised texture classification by integration of multiple texture methods and evaluation windows." Image Vision Computing **25**(7): 1091-1106.

Miller, R. G. (1964). "A trustworthy jackknife." Ann. Math. Statist. **35**(4): 1594-1605.

Minka, T. P. and R. W. Picard (1997). "Interactive Learning Using a Society of Models." Pattern Recognition **30**(3): 565.

MIT. (2002). "VisTex: Visual Texture database." Last access date 01/06/08, from http://vismod.media.mit.edu/vismod/imagery/VisionTexture/vistex.html.

Mojsilovic, A., M. V. Popovic and D. Rackov (2000a). "On the Selection of an Optimal Wavelet Basis for Texture Characterization." IEEE Trans. on Image Processing **9**(12): 2043-2050.

Mojsilovic, A., M. V. Popovic and D. M. Rackov (2000b). "On the selection of an optimal wavelet basis for texture characterization." Image Processing, IEEE Transactions on **9**(12): 2043-2050.

Moore, G. E. (1965). "Cramming more components onto integrated circuits." Electronics **38**(8).

Muller, H. (2004). "A review of content-based image retrieval systems in medical applications—clinical benefits and future directions." International Journal of Medical Informatics **73**(1): 1-23.

Muller, H., S. Marchand-Maillet and T. Pun (2002). The truth about corel - evaluation in image retrieval. In Proceedings of CIVR.

Muneeswaran, K., L. Ganesan, S. Arumugam and K. R. Soundar (2005). "Texture classification with combined rotation and scale invariant wavelet features." Pattern Recognition **38**(10): 1495-1506.

Nabti, M., L. Ghouti and A. Bouridane (2006). "Multiscale edge detection using wavelet maxima for iris localization." IET Conference Publications **2006**(CP522): 62-67.

Nandi, A. V. and R. M. Banakar (2008). Throughput Efficient Parallel Implementation of SPIHT Algorithm. International Conference on VLSO Design (VLSID 2008). Hyderabad, India, IEEE: 718-725.

Nasser, Y., Z. Meral, zsoyoglu and Gultekin (1994). A framework for feature-based indexing for spatial databases. Proceedings of the 7th international conference on Scientific and Statistical Database Management. Charlottesville, Virginia, IEEE Computer Society.

Nastar, C. (1998). Surfimage: a flexible content-based image retrieval system. ACM Multimedia 98, Bristol, UK.

Natsev, A., R. Rajeev and K. Shim (2004). "WALRUS: a similarity retrieval algorithm for image databases." Knowledge and Data Engineering, IEEE Transactions on **16**(3): 301-316.

Neumann, D. and K. R. Gegenfurtner (2006). "Image retrieval and perceptual similarity." ACM Trans. Appl. Percept. **3**(1): 31-47.

Nguyen, T. T. and S. Oraintara (2005). "Multiresolution direction filterbanks: theory, design, and applications." Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on] **53**(10): 3895-3905.

Niblack, W., R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos and G. Taubin (1993). The QBIC project: querying images by content using color, texture, and shape. Storage and Retrieval for Image and Video Databases. San Jose, CA.

Niemann, H. (1981). Pattern Analysis, Springer-Verlag.

nVidia. (2008). "nVidia CUDA platform." Last access date 01/06/08, from http://www.nvidia.com/cuda.

O'Toole, A. J., H. Abdi, F. Jiang and P. J. Phillips (2007). "Fusing Face-Verification Algorithms and Humans." IEEE Trans. on Systems, Man. and Cybernetics **37**(5): 1149-1155.

Ogle, V. E. and M. Stonebraker (1995). "Chabot: Retrieval from a Relational Database of Images." IEEE Computer **28**(9): 40-48.

Ohta, Y. I. and T. Kanade (1980). "Color Information for Region Segmentation." Computer Graphics and Image Processing **13**: 222-241.

Ojala, T., T. Maenpaa, M. Pietikainen, J. Viertola, J. Kyllonen and S. Huovinen (2002). Outex - New framework for empirical evaluation of texture analysis algorithms. Proc. 16th International Conference on Pattern Recognition, Quebec, Canada.

Ojala, T., T. Maenpaa, M. Pietikainen, J. Viertola, J. Kyllonen and S. Huovinen. (2008). "Outex Image Database." Last access date 01/06/08, from http://www.outex.oulu.fi.

Ojala, T. and M. Pietikainen (2002). "Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns." IEEE TRans. on Pattern Analysis and Machine Intelligence **24**(7): 971-987.

Ojala, T., M. Pietikainen and e. al. (2002a). "Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns." IEEE TRans. on Pattern Analysis and Machine Intelligence **24**(7): 971-987.

Ojala, T., M. Pietikainen and D. Harwood (1996). "A comparative study of texture measures with classification based on feature distributions." Pattern Recognition **29**: 51-59.

Ojala, T., M. Pietikainen and T. Maenpaa (2002b). "Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns." Pattern Analysis and Machine Intelligence, IEEE Transactions on **24**(7): 971-987.

Oliveira, M. C., W. Cirne and P. M. de Azevedo Marques (2007). "Towards applying content-based image retrieval in the clinical routine." Future Generation Computer Systems **23**(3): 466-474.

Opera. (2008). "Corporate web-site." Last access date 01/06/08, from http://opera.com.

Ortega-Binderberger, M., S. Mehrotra, K. Chakrabarti and K. Porkaew (2000). WebMARS: A multimedia search engine. Proceedings of the SPIE Electronic Imaging 2000: Internet Imaging San Jose, CA, SPIE.

Ou, G. and Y. L. Murphey (2007). "Multi-class pattern classification using neural networks." Pattern Recognition **40**(1): 4-18.

Pan, W., T. D. Bui and C. Y. Suen (2008). "Rotation invariant texture classification by ridgelet transform and frequency-orientation space decomposition." Signal Processing **88**(1): 189-199.

Pao, M. L. and M. Lee (1989). Concepts of Information Retrieval. Englewood, CO, Libraries Unlimited, Inc. .

Park, Y. K. and J. K. Kim (2007). "Fast adaptive smoothing based on LBP for robust face recognition." IET Electronics Letters **43**(24).

Pentland, A. (1984). "Fractal-based Description of Natural Scenes." IEEE TRans. on Pattern Analysis and Machine Intelligence **6**(6): 661-674.

Pentland, A., R. Picard and S. Sclaroff (1994). Photobook: Content-based Manipulation of Image Databases. SPIE Storage and Retrieval for Image and Video Databases II, San Jose, CA, SPIE.

Photosynth. (2008). "Microsoft Photosynth web-site." Last access date 02/06/08, from http://labs.live.com/photosynth/.

Pi, M. H., C. S. Tong, S. K. Choy and H. Zhang (2006). "A Fast and Effective Model for Wavelet Subband Histograms and Its Application in Texture Image Retrieval." Image Processing, IEEE Transactions on **15**(10): 3078-3088.

Pinto Ellas, R., O. O. V. Villegas and M. Lopez Sanchez (2006). Wavelet Lossy Image Coding with Edge and Texture Preserving Using a Modified SPIHT. Computational Intelligence and Security, 2006 International Conference on.

Porter, R. (1997). "Robust rotation-invariant texture classification: wavelet, Gabor filter and GMRF based schemes." Vision, Image and Signal Processing, IEE Proceedings **144**(3): 180-188.

Porter, R. and N. Canagarajah (1997). "Robust rotation-invariant texture classification: wavelet, Gabor filter and GMRF based schemes." Vision, Image and Signal Processing, IEE Proceedings- **144**(3): 180 -188.

Poulakidas, A. S., A. Srinivasan, O. Egecioglu, O. Ibarra and T. Yang (1997). A compact storage scheme for fast wavelet-based subregion retrieval.

Proceedings of Third Annual International Computing and Combinatorics Conference, Shanghai, China.

Pujari, A. K. (2001). Data mining techniques. Hyderabad; [Great Britain], Universities Press.

Qahwaji, R. and T. Colak (2007). "Automatic Short-Term Solar Flare Prediction Using Machine Learning and Sunspot Associations." Solar Physics **241**: 195-211.

Qi, F. and D. Shen (1994). "Wavelet Transform Based Rotation Invariant Feature Extraction in Object Recognition." Proc. of Intl. Symp. on Information Theory & Its Applications: 221-224.

Qiao, X., F. Murtagh, P. Walsh, P. A. M. Basheer, A. Long and D. Crookes (2004). ""Virtual sieve": content-based image retrieval in automated grading of engineering materials." IEE Conference Publications **2004**(CP506): 449-453.

Rabbani, M. and R. Joshi (2002a). "An overview of the JPEG2000 still image compression standard." Signal Processing: Image Communication **17**(1): 1-46.

Rabbani, M. and R. Joshi (2002b). "An overview of the JPEG2000 still image compression standard." Signal Processing: Image Communication **17**(1): 1-46.

Rashkovskiy, O., L. Sadovnik and N. P. Caviris (1994). Scale, rotation and shift invariant wavelet transform. Optical Pattern Recognition V. Orlando, FL.

Ros, J., C. Laurent and G. Lefebvre (2006). A Cascade of Unsupervised and Supervised Neural Networks for Natural Image Classification Lecture Notes in Computer Science: Image and Video Retrieval. Heidelberg, Springer Berlin. **4071/2006:** 92-101.

Rosenbaum, R. and H. Schumann (2006). JPEG2000-based viewer guidance for mobile image browsing. Multi-Media Modelling Conference Proceedings, 2006 12th International Germany.

Rosenfeld, A. and A. Kak (1982). Digital Picture Processing, vol.1, Academic Press.

Rosenfeld, A. and J. Weszka (1980). Picture Recognition. Digital Pattern Recognition. K. Fu, Springer-Verlag: 135-166.

Rui, Y. and T. S. Huang (2000). Optimizing learning in image retrieval. IEEE International Conference on Computer Vision and Pattern Recognition. Hilton Head, SC, USA.

Rui, Y., T. S. Huang and S.-F. Chang (1999). "Image Retrieval: Current Techniques, Promising Directions And Open Issues." Journal of Visual Communication and Image Representation **10**: 39-62.

Saber, E. and A. M. Tekalp (2006). "Integration of color, edge, shape, and texture features for automatic region-based image annotation and retrieval." Journal of Electronic Imaging **7**(3): 684-700.

Saha, S. K., A. K. Das and B. Chanda (2007). "Image retrieval based on indexing and relevance feedback." Pattern Recognition Letters **28**(3): 357-366.

Said, A. (2006). Image Compression with Set Partitioning in Hierarchical Trees - web site.

Said, A. and A. Pearlman (1996a). "A New, Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees." IEEE Trans. on Circuits and Systems for Video Technology **6**: 242-250.

Said, A. and W. A. Pearlman (1996b). "A New, Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees." IEEE Trans. on Circuits and Systems for Video Technology **6**: 243-250.

Salton, G. (1989). Automatic text processing. Reading, Mass., Addison-Wesley.

Sam-Deuk, K. and S. Udpa (2000). "Texture classification using rotated wavelet filters." Systems, Man and Cybernetics, Part A, IEEE Transactions on **30**(6): 847-852.

Sameer, S. and M. Sharma (2001). Texture Analysis Experiments with Meastex and Vistex Benchmarks. Proceedings of the Second International Conference on Advances in Pattern Recognition, Springer-Verlag.

Santa-Cruz, D. and T. Ebrahimi (2000). An analytical study of JPEG 2000 functionalities. Proc. of the IEEE International Conference on Image Processing (ICIP), Vancouver, Canada.

Santa-Cruz, D., R. Grospois and T. Ebrahimi (2002). "JPEG 2000 performance evaluation and assesment." Signal Processing: Image Communication **17**(1): 113-130.

Sastry, C. S., A. K. Pujari, B. L. Deekshatulu and C. Bhagvati (2004). "A wavelet based multiresolution algorithm for rotation invariant feature extraction." Pattern Recognition Letters **25**(16): 1845-1855.

Schaefer, G. (2004). "JPEG2000 vs. JPEG from an image retrieval point of view." Image Processing, 2004. ICIP'04. 2004 International Conference on **1**.

Schalkoff, R. (1989). Digital Image Processing and Computer Vision, ch.6, John Wiley & Sons.

Scharcanski, J. (2007). "A Wavelet-Based Approach for Analyzing Industrial Stochastic Textures With Applications." Systems, Man and Cybernetics, Part A, IEEE Transactions on **37**(1): 10-22.

Schwenk, H. and Y. Bengio (1997). Adaboosting neural networks: application to on-line character recognition. Proc. Int. Conf. Artificial Neural Networks (ICANN'97).

Sclaroff, S., L. Taycher and M. La Cascia (1997). Image rover: A content-based image browser for the World Wide Web. Proceedings of the IEEE Workshop on Content-based Access of Image and Video Libraries, San Juan, Puerto Rico.

Seloff, G. A. (1990). "Automated access to NASA-JSC image archives." Library Trends **38**(4): 682-696.

Selvan, S. and S. Ramakrishnan (2007). "SVD-Based Modeling for Image Texture Classification Using Wavelet Transformation." Image Processing, IEEE Transactions on **16**(11): 2688-2696.

Sengur, A. (2008). "Wavelet transform and adaptive neuro-fuzzy inference system for color texture classification." Expert Systems with Applications **34**(3): 2120-2128.

Serra, J. (1982). Image Analysis and Mathematical Morphology, Academic Press.

Servetto, S. D., K. Ramchandran and M. T. Orchard (1999). "Image coding based on a morphological representation." IEEE Trans. on Image Processing **8**: 1161-1174.

Shapiro, J. M. (1993). "Embedded image coding using zerotrees of wavelet coefficients." IEEE Trans. on Signal Processing **41**(12): 3445-3462.

Shih, J.-L., C.-H. Lee and C.-S. Yang (2007). "An adult image identification system employing image retrieval technique." Pattern Recognition Letters **28**(16): 2367-2374.

Shneier, M. and M. Abdel-Mottaleb (1996). "Exploiting the JPEG compression scheme for image retrieval." IEEE Transactions on Pattern Analysis and Machine Intelligence **18**(8): 849-53.

SiliconImaging. (2008). "Corporate web-site." Last access date 01/06/08, from http://www.siliconimaging.com.

Simoncelli, E. and J. Portilla (1998). Texture Characterization via Joint Statistics of Wavelet Coefficient Magnitudes. IEEE International Conference on Image Processing, Chicago, IEEE Computer Society.

Smeulders, A. W. M., M. Worring, S. Santini, A. Gupta and R. Jain (2000). "Content-based Image Retrieval at the End of the Early Years." IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(12): 1349-1380.

Smith, J. and S. F. Chang (1994). Quad-tree segmentation for texture-based image query. Proceedings of the second ACM international conference on Multimedia. San Francisco, California, United States, ACM.

Smith, J. R. (1997). Integrated Spatial and Feature Image Systems: Retrieval, Analysis and Compression, Columbia University.

Smith, J. R. (2001). "Quantitative assessment of image retrieval effectiveness." Journal of the American Society for Information Science and Technology **52**(11): 969-979.

Smith, J. R. and S.-F. Chang (1996). VisualSEEk: a fully automated content-based image query system. ACM Multimedia 96, Boston, MA, ACM.

Smith, J. R. and C. Shih-Fu (1994). Transform features for texture classification and discrimination in large image databases. Proceedings of 1st International Conference on Image Processing. Austin, TX.

Smith, S. W. (1999). The Scientist and Engineer's Guide to Digital Signal Processing Second Edition. San Diego, California Technical Publishing.

Smith, S. W. (2002). Digital Signal Processing: A Practical Guide for Engineers and Scientists, Newnes.

Snavely, N., S. M. Seitz and R. Szeliski (2006). Photo tourism: Exploring photo collections in 3D. SIGGRAPH Conference Proceedings. New York, NY, USA, ACM Press: 835--846.

Spink, A. and B. J. Jansen (2006). "Searching multimedia federated content Web collections." Online Information Review **30**(5): 485-495.

Stanchev, P. L., D. Green Jr. and B. Dimitrov (2003). "High level color similarity retrieval." Int. J. Inf. Theories Appl. **10**(3): 363-369.

Stokes, J. (2008). "Understanding Moore's Law." arstechnica Last access date 02/06/08, from http://arstechnica.com/articles/paedia/cpu/moore.ars/1.

Strang, G. and T. Nguyen (1996). Wavelets and filter banks, Wellesley-Cambridge Press.

Su, C. K., H. C. Hsin and S. F. Lin (2005). "Wavelet tree classification and hybrid coding for image compression." IEE Proceedings - Vision, Image, and Signal Processing **152**(6): 752-756.

Sung, T.-Y. and H.-C. Hsin (2007). "An Efficient Rearrangement of Wavelet Packet Coefficients for Embedded Image Coding Based on SPIHT Algorithm." IEICE Trans Fundamentals **E90-A**(9): 2014-2020.

Tabesh, A., A. Bilgin, K. Krishnan and M. W. Marcellin (2005). "JPEG2000 and Motion JPEG2000 content analysis using codestream length information." Data Compression Conference, 2005. Proceedings. DCC 2005: 329-337.

Tan, X. and B. Triggs (2007). Fusing Gabor and LBP Feature Sets for Kernel-Based Face Recognition. Analysis and Modeling of Faces and Gestures: 235-249.

Tardif, P. M. and A. Zaccarin (1997). "Multiscale autoregressive image representation for texture segmentation." Non-linear Image Processing **3026**: 327-337.

Tegolo, D. (1994). "Shape analysis for image retrieval." Proc. of SPIE: Storage and Retrieval for Image and Video Databases II **2185**: 59-69.

Teynor, A., W. Muller and W. Kowarschick (2006a). Compressed Domain Image Retrieval Using JPEG2000 and Gaussian Mixture Models, SPRINGER-VERLAG.

Teynor, A., W. Muller and W. Kowarschick (2006b). Compressed Domain Image Retrieval Using JPEG2000 and Gaussian Mixture Models. Lecture Notes in Computer Science: Visual Information and Information Systems, SPRINGER-VERLAG. **3736:** 132-143.

Thomson, D. J. (2007). Jackknifing Multitaper Spectrum Estimates. IEEE Signal Processing Magazine, IEEE. **July 2007:** 20-30.

Tipping, M. E. (2001). "Sparse bayesian learning and the relevance vector machine." J. Mach. Learn. Res. **1**: 211-244.

Tjondronegoro, D. and A. Spink (2008). "Web search engine multimedia functionality." Information Processing & Management **44**(1): 340-357.

Tsai, C.-F., K. McGarry and J. Tait (2006). "CLAIRE: A modular support vector image indexing and classification system." ACM Trans. Inf. Syst. **24**(3): 353-379.

Tsui, J. (2004). Digital Techniques for Wideband Receivers, 2nd edition, SciTech Publishing.

Tuceryan, M. and A. K. Jain (1993). Texture Analysis. Handbook of Pattern Recognition and Computer Vision. C. H. e. a. Chen, World Scientific: 235-276.

Tuceryan, M. and A. K. Jain (1998). Texture Analysis. Handbook of Pattern Recognition and Computer Vision. C. H. Chen, L. F. Pau and P. S. P. Wang, World Scientific: 235-276.

Tusk, C., K. Koperski, S. Aksoy and G. Marchisio (2003). Automated feature selection through relevance feedback. Geoscience and Remote Sensing Symposium, 2003. IGARSS apos;03. Proceedings. , IEEE International.

Unay, D., A. Ekin, M. Cetin, R. Jasinschi and A. Ercil (2007). Robustness of Local Binary Patterns in Brain MR Image Analysis. Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE Lyon, France 2098 - 2101

Unser, M. (1995). "Texture classification and segmentation using wavelet frames." IEEE Trans. on Image Processing **4**: 1549-1560.

Valkealahti, K. and E. Oja (1998). "Reduced multidimensional co-occurrence histograms in texture classification." IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(1): 90-4.

Van de Wouwer, G., S. Livens, P. Scheunders and D. Van Dyck (1997). Color texture classification by wavelet energy correlation signatures. Proceedings of ICIAP 97. 9th International Conference on Image Analysis and Processing. Florence, Italy. IAPR. 17-19 Sept. 1997.

Van de Wouwer, G., P. Scheunders, S. Livens and D. Van Dyck (1999a). "Wavelet correlation signatures for color texture characterization." Pattern Recognition **32**(3): 443-51.

Van de Wouwer, G., P. Scheunders and D. Van Dyck (1998). Rotation-invariant texture characterization using isotropic wavelet frames. Proceedings Fourteenth International Conference on Pattern Recognition. Brisbane, Qld., Australia. 16-20 Aug. 1998.

Van de Wouwer, G., P. Scheunders and D. Van Dyck (1999b). "Statistical texture characterization from discrete wavelet representations." IEEE Transactions on Image Processing **8**(4): 592-598.

van der Starre, J. H. E., Ed. (1995). Ceci n'est pas une pipe: indexing of images. Multimedia Computing and Museums: selected papers from the 3rd International conference on Hypermedia and Interactivity in Museums. San Diego, California, Archives and Museum Informatics.

Vapnik, V. N. (1995). The Nature of Statistical Learning Theory. New York, Springer-Verlag.

Vasconcelos, N. (2007). "From Pixels to Semantic Spaces Advances in Content-based Image Retrieval." IEEE Computer(July 2007): 20-26.

Vassilakopoulos, Y., K. Manolopoulos and K. Economou (1993). "Overlapping quadtrees for the representation of similar images." Image and Vision Computing **11**: 257-262.

Viola, P. and M. Jones (2001). Rapid object detection using a boosted cascade of simple features. Int. Conf. Computer Vision and Pattern Recognition: 511-518.

Virage. (2008). "Corporate web-site." Last access date 01/06/08, from http://www.virage.com.

Voulgaris, G. (2008). "Generic CBIR Sample Image Database - University of Glamorgan Digital Imaging Research Group." Last access date 2/6/2008, from http://voulgaris.net/phd/cbir-database.rar.

Voulgaris, G. and J. Jiang (2001a). A Combined Texture-based Image Indexing and Compression System. Proc. of EUROIMAGE 2001 International Conference on Augmented, Virtual Environments and 3D Imaging, Mykonos : Greece, Informatics and Telematics Institute-CERTH.

Voulgaris, G. and J. Jiang (2001b). Low Complexity Content Based Image Indexing in Wavelets Domain. Proc. of PREP 2001, Keele : UK, EPSRC.

Voulgaris, G. and J. Jiang (2001c). Texture-based Image Retrieval in Wavelets Compressed Domain. Proc. of IEEE International Conference on Image Processing, Thessaloniki : Greece, IEEE Signal Processing Society.

Voulgaris, G. and J. Jiang (2001d). Wavelet-based Image Indexing and Retrieval in Compressed Domain. Proc. 1st Annual Doctoral Seminar, Treforest : Wales, Pont Dysgu.

Voulgaris, G. and J. Jiang (2002a). Combined Colour and Texture Information Extraction in Wavelets Domain For Image Indexing. Proc. of PREP 2002, Nottingham : UK, EPSRC.

Voulgaris, G. and J. Jiang (2002b). Faster Distance Calculation in Image Retrieval Systems. IEEE International Conference on Consumer Electronics, Los Angeles: USA, IEEE Consumer Electronics Society.

Voulgaris, G. and J. Jiang (2002c). Quadtree-based image indexing in wavelets compressed domain. Proc. of 20th Annual Eurographics UK, Leicester : UK, IEEE Computer Society Press.

Voulgaris, G., A. J. Ware and M. Reddy (2002). Wavelet-based Imaging and Applications. Proc. of 2nd Annual Doctoral Seminar, Treforest : Wales, Pont Dysgu.

Wang, H. and S. F. Chang (1996). Adaptive image matching in the subband domain. Visual Communications and Image Processing '96. Orlando, FL.

Wang, J. Z. (2001). "Wavelets and Imaging Informatics: A Review of the Literature." Journal of Biomedical Informatics **34**(2): 129-141.

Wang, J. Z., L. Jia and G. Wiederhold (2001). "SIMPLIcity: semantics-sensitive integrated matching for picture libraries." Pattern Analysis and Machine Intelligence, IEEE Transactions on **23**(9): 947-963.

Wang, J. Z., J. Li, R. M. Gray and G. Wiederhold (2001). "Unsupervised Multiresolution Segmentation for Images with Low Depth of Field." IEEE Trans. Pattern Analysis and Machine Intelligence **23**(1): 85-91.

Wang, J. Z., G. Wiederhold, O. Firschein and S. Xin Wei (1998). "Content-based image indexing and searching using Daubechies' wavelets." International Journal on Digital Libraries **1**(4): 311-328.

Wang, Z., A. C. Bovik, H. R. Sheikh and E. P. Simoncelli (2004). "Image quality assessment: from error visibility to structural similarity." Image Processing, IEEE Transactions on **13**(4): 600-612.

Weber, R., H. J. Schek and S. Blott (1998). A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. Proceedings of the 24rd International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc.

Wei, H., B. Zhao and P. He (2007). Hyperspectral image compression using SPIHT based on DCT and DWT. MIPPR 2007: Multispectral Image Processing, Wuhan, China, SPIE.

Wei, L., Y. Yang, R. M. Nishikawa and Y. Jiang (2005). "A Study on Several Machine-Learning Methods for Classification of Malignant and Benign Clustered Microcalcifications." IEEE Trans. on Medical Imaging **24**(3): 371-380.

Weszka, J. and C. Deya (1976). "A Comparative Study of Texture Measures for Terrain Classification." IEEE Trans. on System, Man and Cybernetics **6**: 269-285.

Whitaker, J. C., Ed. (2005). The Electronics Handbook Second Edition. Boca Raton, CRC Press.

Wilson, B. A. and M. A. Bayoumi (2003). "A computational kernel for fast and efficient compressed-domain calculations of wavelet subband energies." Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on [see also Circuits and Systems II: Express Briefs, IEEE Transactions on] **50**(7): 389-392.

Wilson, R., A. D. Calway, H. Tao and P. R. Meulemans (1995). Feature extraction for low bit rate image coding using a generalized wavelet transform, University of Bristol.

Wu, J. K., M. S. Kankanhalli, J.-H. Lim and D. Hong (2002). Perspectives on content-based multimedia systems. New York, Kluwer Academic Publishers.

Xing, X., L. Hao, M. Wei-Ying and Z. Hong-Jiang (2006). "Browsing large pictures under limited display sizes." Multimedia, IEEE Transactions on **8**(4): 707-715.

Yahoo! (2008). "Yahoo! image search." Last access date 01/06/08, from http://images.search.yahoo.com/.

Yang, S. H. and P. F. Cheng (2007). "Robust Transmission of SPIHT-Coded Images Over Packet Networks." Circuits and Systems for Video Technology, IEEE Transactions on **17**(5): 558-567.

Yu, C., B. Cui, S. Wang and J. Su (2007). "Efficient index-based KNN join processing for high-dimensional data." Inf. Softw. Technol. **49**(4): 332-344.

Zhang, C. and Y.-h. Cheng (2007). A Hybrid Template Match Approach Based on Wavelet Analysis and Threshold Segmentation for Detecting Tire Surface Wear. Control and Automation, 2007. ICCA 2007. IEEE International Conference on.

Zhang, J. and T. Tan (2002). "Brief review of invariant texture analysis methods." Pattern Recognition **35**: 735-747.

Zhang, R. and Z. Zhang (2007). "Effective Image Retrieval Based on Hidden Concept Discovery in Image Database." Image Processing, IEEE Transactions on **16**(2): 562-572.

Zhao, G. and M. Pietikainen (2007). "Dynamic Texture Recognition Using Local Binary Patterns with an Application to Facial Expressions." Pattern Analysis and Machine Intelligence, IEEE Transactions on **29**(6): 915-928.

Zhi-Hua, Z., C. Ke-Jia and D. Hong-Bin (2006). "Enhancing relevance feedback in image retrieval using unlabeled data." ACM Trans. Inf. Syst. **24**(2): 219-244.

Zhong, D. and I. Defee (2007). "Performance of similarity measures based on histograms of local image feature vectors." Pattern Recognition Letters **28**(15): 2003-2010.

Zhou, X. S. and T. S. Huang (2003). "Relevance feedback in image retrieval: A comprehensive review." Multimedia Systems **8**(6): 536-544.

Zusne, L. (1965). "Moments of Area and of the Perimeter of Visual Form as Predictors of Discrimination Performance." Journal of Experimental Psychology **69**(1965): 213-220.

Zusne, L. (1970). Visual Perception of Form. New York, Academic Press.

Zuyuan, W. and R. Boesch (2007). "Color- and Texture-Based Image Segmentation for Improved Forest Delineation." Geoscience and Remote Sensing, IEEE Transactions on **45**(10): 3055-3062.

Chapter 6: Conclusions

6.1 Abstract

Here follows a summary of all research that is documented in this thesis. Conclusions and limitations are discussed giving leads for future work.

6.2 Introduction

The rationale for this research builds upon the overall conclusion that CBIR has not yet reached sufficient performance balance of accuracy, efficiency and speed for wide adoption in practical applications. Attention was drawn to CBIR in transform domain and in particular wavelets domain because of its excellent characteristics for compression and texture extraction applications and the wide adoption in the research community and the industry. Accuracy of texture analysis and in particular the sensory gap has yet to be resolved. Considering the demand for wavelets-based techniques it would be useful to match or exceed the performance balance of similar pixel-based techniques. Speed improvement remains an uncharted territory as is feature extraction directly from the bit-stream of compressed images.

Therefore, the general aim of this research is to provide the means for improving accuracy and speed of content-based image indexing, retrieval and classification in wavelets domain. In particular, this research sought to:

- Suggest new and build upon existing techniques for improving the accuracy of texture extraction in wavelets domain; ideally these should be compatible and capable of being extended to compressed domain.
- Suggest methods for improving the speed of CBIR/CBIC systems;
- Suggest methods for feature extraction directly from the compressed-bit stream with no decompression stage required.

Empirical results and techniques which were proposed in the preceding chapters addressed these research objectives. This chapter brings together the aforementioned techniques and suggests approaches for further extensions of the work.

6.3 Overview of literature survey

Here is a summary of the shortcomings of the existing research identified from the literature survey in Chapter 2 as a link between the preceding chapters and the conclusions and suggested future work presented in the following sections. From this exposition of the published literature, it is possible to address a number of important issues, as follows:

- Major issues related to low-level CBIR performance have not yet been addressed effectively by the existing research. Accuracy and speed remain two of those core issues of image retrieval which require further investigation.
- The similarity between the characteristics of wavelets transformation and those of the HVS has made the former an excellent choice for texture analysis. Yet some pixel-based techniques offer better overall performance.
- Several practical applications of texture analysis are still in demand of efficient texture analysis techniques.
- Rotation invariance of wavelet-based texture descriptors is a "sensory gap" issue which requires further study as pixel domain counterparts seem to provide more efficient solutions. Furthermore, there is a clear gap when it comes to rotation invariant techniques which have the capability to be extended for direct extraction from wavelet-based compressed images.
- Quad-trees are at the heart of the wavelet transform and have been exploited for image compression and pixel-based feature extraction. However, they have not yet been applied as a mechanism for texture feature extraction from wavelet coefficients.
- A general issue of Image Processing systems, which also applies to image indexing and retrieval, is the need for methods that can efficiently extract feature information directly from the compressed bit-stream using standard compression schemes.
- Signature distance calculation is a significant contributing factor to the overall speed of an image retrieval query. Current research on improving the speed of distance calculation is limited, thus leaving a lot of scope for further studies.

6.4 Overview of the contributions; connecting the dots

Bringing the threads of this research together, all the methods and algorithms presented in this work are linked by two underlying fundamental objectives:

a) How to improve content-based image retrieval systems...
b) ... while ensuring that all suggested techniques can be applied to "compressed domain" without compromising accuracy, according to the description given in Sections 1.5 and 2.10.

From a high-level perspective (see Figure 1.7) the techniques can be split into two general categories with respect to the first objective: a) those techniques that attempt to improve accuracy (Chapters 3 & 4) and b) those techniques that attempt to improve speed (Chapter 5). The techniques which focus on accuracy are doing so by improving existing (Localization Grid and Combined Energy-Variance) and introducing new (Quad-trees, LWCP and Unified HVD) algorithms for texture feature extraction in wavelets domain. Finally those techniques can be further broken down according to those that appeared more promising for retrieval applications (Localization Grid & Quad-trees), those that appeared more promising for classification applications (LWCP & QT Patterns) and those focusing on how to solve the problem of rotation invariance (Unified HVD and Combined Energy-Variance). Techniques focusing on speed improvement can be categorized to those focusing on improving the distance calculation (Pre-filtering type A and type B methods) and one technique which extract features directly from native compressed bit-stream (Direct feature extraction method).

With respect to the second fundamental objective all algorithms are designed to work with as few as possible decompression stages involved. Referring to Figure 1.2, the algorithms operate at point C or, in the case of the Direct Feature Extraction technique, at point E, thereby qualifying as working in compressed domain. In order to achieve this, all algorithms employ orthogonal DWT and parameters that are being or can be used by existing compression codecs such as JPEG2000 (Rabbani and Joshi 2002b) and SPIHT (Said and Pearlman 1996b). In fact, most of the algorithms are based on Significance Map Encoding which is part of the quantization stage of a wavelet codec, therefore being closer to the compressed domain than point C in Figure 1.2.

Moving one step further, the proposed algorithms might be combined and integrated with other required subcomponents in an ideal scenario in order to create the "ultimate" CBIR system; a system capable of extracting content information quickly and accurately, irrespective of whether the query image is rotated and searching directly from compressed image databases without performing any decompression. A technique might be used in order to establish which of the different feature extraction techniques performs best in a specific query, such as the automated feature selection through relevance feedback described in (Tusk, Koperski et al. 2003). For distance calculation, provided that the L2-distance is used and that the database is static, the low-cost Pre-filtering type A could be used as a first stage reduction of the database to ½ of the original size (since this reduction was found not to affect accuracy) and then pre-filtering type B, which is uncorrelated to feature performance, could be used for further reduction. If the database is dynamic then the Direct Feature extraction method could be used for improving the speed and the processing cost.

The optimistic statements above however can only be materialized under two basic assumptions, which are: to overcome the limitations of the techniques presented in this work and to overcome all limitations and issues of the CBIR status quo which are not directly related to this work, on which Chapters 1 and 2 elaborated. Therefore, here follows a list of the main identified issues that have to be resolved in order for the algorithms presented in this work to move closer to finding their way into the "ultimate" CBIR:

- Rotation invariance is not the only sensory-gap issue which could affect the comparison of two images with similar content. Other important variables include: linear transformations (rotation, scaling or shear), translations (or "shifts"), global illumination differences, internal illumination unevenness and contrast. Although the importance of each of those variables depends on the quality and consistency of the image database and of the query image, the "ultimate" CBIR system should either include features which accommodate invariance for all these or it should be able to select different features for different cases manually or automatically.

- The assertions made with regards to the grid sizes of the Localization Grid have only considered square aspect ratios and even distribution of cells. Not all images are square so the behaviour of the algorithm needs to be studied for images with irregular or at least parallelogram aspect ratios. Uneven cell size distribution should also be studied in order to offer more fine tuned localization grids which would include more detail in image areas with higher significance.
- Texture is not the panacea of features for comparing two images for content similarity. Other features both low-level such as colour or shape and perhaps higher level such as objects should be incorporated in order to improve the accuracy but also to provide a richer set of options to the user for query specification.
- The retrieval and classification metrics used in this work were chosen based on ease of implementation and due to wide adoption by the image processing research community. Although these metrics provided a first level validation and proof of concept for the algorithms newer more advanced methodologies such as those discussed in Sections 2.8 and 3.5.2.1 should be tested in an attempt to fine tune accuracy and stabilize the performance of the classifier.
- All algorithms should be tested against further benchmarks and larger real-life data sets in order to offer more concrete and complete validation of their performance.
- The advantage of the Direct Feature extraction technique largely depends on the mechanics of the SPIHT algorithm (Said and Pearlman 1996b). This is also its disadvantage since a vast existing volume of imagery is compressed using the original JPEG compression codec and less often the JPEG2000 compression codec. To this end, it would be interesting to investigate if and how the methodology of Direct Feature extraction can be adapted to JPEG2000 and establish a delta between the performances. Furthermore, another study could analyze the total processing cost of the feature extraction techniques presented in this work when combined with either a JPEG2000 or SPIHT based decoder whilst attempting to exploit the decoders for the specific feature extraction operations following the paradigm of the Direct Feature extraction algorithm.

Further to perfecting and adapting the proposed algorithms several other issues should be considered when building the ultimate CBIR system such as the user interface and the query specification mechanisms, methods to address the semantic gap and optimizing the interactions with the image databases. As further analysis of each one of these subjects easily fills several theses the reader is referred for more information to one of the excellent reviews suggested in Section 2.2.

Starting by indicating the common denominators of the methods presented in this thesis, this section discussed the possibilities and the major issues associated with the integration of the presented techniques in a real CBIR system. In the following sections the issues that are related to the improving the design and performance of those techniques are highlighted along with further suggestions for extending this work.

6.5 Overall conclusions

From the outcomes of this research, a number of conclusions based on empirical analysis become apparent:

- Savings in processing cost, and thus in the time required for retrieval or classification, can be attained by fine tuning feature characterization algorithms so that only the most significant DWT data are analyzed.
- Orthogonal DWT can be used in order to achieve accurate texture extraction for retrieval and classification. Intra-subband local analysis is a particularly effective approach for improving the accuracy of traditional statistical-based methods. There is still room for improvement, however the indications are solid.
- In cases where pre-calculation of signatures is acceptable, improvement in the speed of retrieval and classification can be achieved by proper manipulation of the signature database. It is possible to restrict the relevant data set, by proper mathematical or morphological analysis of the feature signatures.
- With the proper combination of compression codec and feature extraction technique, it is possible to perform accurate on-line feature extraction directly from compressed coefficients.

Using as a basis the conclusions drawn in this chapter, the following section provides direct answers to the original aims set at the onset of this research project.

Suggest new and build upon existing techniques for improving the accuracy of texture extraction in wavelets domain; ideally these should be compatible and capable of being extended to compressed domain.

Four different techniques were introduced, addressing wavelet-based feature characterization from three different perspectives. Empirical results prove that it is feasible to design such systems and to achieve sufficient accuracy. The capability of the suggested algorithms was tested on both retrieval and classification. Quad-tree Patterns and LWCP are more suitable for texture classification, as they exploit the pattern structures contained in DWT subband in more detail than the remaining algorithms. The characteristic of the Localization Grid and the Quad-tree algorithms to convey both texture and coefficient distribution information, results in superior performance in retrieval experiments.

The scope was extended to the more specialized case of rotation-invariance. To address the sensory gap issue of rotation invariance, the concept of Wavelet Isometric Operators is introduced. This refers to an approximation model of the changes that the wavelet decomposition tree undergoes as an image gets rotated. Empirical findings indicate that the model can form a solid basis for the implementation of rotation-invariant texture extraction algorithms. In particular, Unified HVD is a promising texture characterization technique that performs competitively to existing algorithms in both retrieval and classification.

Suggest methods for improving the speed of CBIR/CBIC systems

Coefficient significance, as it is defined in Significance Map encoding, is tightly connected to wavelet-based feature characterization. For all feature extraction algorithms that were presented throughout this research, coefficient significance and the multi-resolution representation of DWT are used as the basis to trim down the data volume that is processed by the

feature extraction engine. Evidently, savings in processing time can be achieved without compromising accuracy by proper pre-filtering of the DWT decomposition tree, depending on which feature extraction algorithm is used.

Moreover, two methods were introduced targeting the feature signature distance calculation process. The morphological analysis of feature histogram signatures and the boundary conditions techniques accomplish speed improvement without any loss in accuracy. Empirical results indicate that considerable improvement in speed can be gained by using the morphological analysis pre-filtering and boundary conditions pre-filtering techniques.

Suggest methods for feature extraction directly from the compressed-bit stream with no decompression stage required.

A novel technique for feature extraction was introduced which extracts all necessary data for feature extraction from the compressed bit-stream using the well known SPIHT compression codec. The algorithm supplements the logic of SPIHT in order to automatically determine location and significance of DWT coefficients without any decompression. The presented technique incorporates both texture and shape characterization whilst performing adequately both in terms of retrieval and classification accuracy.

6.6 Contribution to knowledge

The scope of this research covers the areas of content-based image retrieval in wavelets compressed domain and, more specifically, the performance balance of accuracy, efficiency and speed.

Accuracy

- New algorithms for texture and combined texture/shape feature extraction in wavelets domain were proposed. Aiming at existing wavelet-based compressed databases, the algorithms offer excellent accuracy when compared to existing designs, whilst ensuring compatibility with wavelet-based compression codecs. In order to improve the feature characterization

ability, the focus was thus on the extraction technique rather than the wavelet decomposition parameters.

- A new model for rotation invariant feature characterization using orthogonal wavelets was introduced. The proposed system addresses the issue of directionality in orthogonal wavelet decomposition and its effect in feature extraction. Empirical results suggest that the Wavelet Isometry approximation model can be used as the basis in order to design rotation invariant feature extraction systems. Prototype implementations were tested on retrieval and classification offering very promising results.

Efficiency & Speed

- All algorithms were tested against dependence on information availability in order to establish whether higher efficiency can be achieved by discarding part of the available data without losses in accuracy. Notably, as source data were gradually reduced, the rate of reduction in accuracy varied between different algorithms. In some cases considerable savings could be achieved without compromise in accuracy. A general conclusion drawn from empirical analysis is that, subband importance determines to a big extent whether or not its information content must be used for feature characterization.
- Algebraic and morphological analysis techniques were introduced designed to speed up the signature matching process. The proposed techniques offer a new perspective on speeding up the retrieval and classification processes by manipulation of the signature databases in order to pre-screen part of the signatures, thus performing the full distance calculation process on a smaller data set. Empirical results indicate that considerable savings can be achieved using either technique.
- The last contribution was a novel technique for feature extraction by direct manipulation of the compressed bit stream. The method addressed both issues of: a) signature storage requirements and, b) all drawbacks associated with database pre-processing for feature signature generation in CBIR/CBIC systems. Although, the accuracy of the technique must be improved in order to be directly competitive with some state-of-the-art algorithms, the results are very promising and clearly justify further study.

6.7 Suggestions for future work

Throughout this work it has been clear that there is vast scope for further work in the field of content-based retrieval and classification in wavelets compressed domain. In the remaining sections of this chapter follows an outline of derived objectives that the author would wish to further explore in the future. Future work suggestions follow two orientations: a) augmentation of the work presented in this thesis and b) exploration of new fields and applications.

Augmentation of the work presented in this thesis

Although the observations that were made throughout this work are important, there are still unexplored strands of the presented ideas. For instance, there are other external factors that affect the visual perception of content of real-life images, for example, illumination distortions, grey-scale variance and affine transformations (including shift, shear and scaling). The presented algorithms should be thoroughly tested against those issues and new techniques should be investigated where appropriate in order to encompass such invariance capabilities. This could require the use of differents form for representing and different techniques for comparing the feature vectors.

Specifically in terms of scale invariance, although some of the presented techniques take advantage of the DWT decomposition in order to compare textures at different scales, this limits the ability to detect differences to those that are different in a dyadic form (*i.e.,* two- four- eight- etc. times larger). Since the differences between scales in natural images may appear in any ratio a more thorough multi-scale invariant scheme has to be incorporated. A good starting point for such a technique is the LWCP algorithm which can easily be adapted by using a descriptor with larger radius.

Furthermore, the techniques that are presented in this thesis can be used as the basis for further studies on alternative configurations. Examples of this are the implementation of Quad-tree Patterns of more than 2 levels, Localization Grid pre-query testing in order to establish which grid size gives the best accuracy (*e.g.* by establishing a ratio of *[subband importance][spatial distribution of importance within the subband]*) and LWCP of various radiuses and components.

The image sizes that were used for testing throughout this work are at maximum 512x512 pixels. This was a choice based on convenience in the implementation and on processing requirements. In printed-press quality environments, high dot-per-inch resolution is of primary importance; therefore image sizes can easily exceed millions of pixels. It would be interesting to investigate the behaviour of retrieval and classification algorithms using such datasets where the granularity at pixel level becomes a minor issue and only macroscopic perceptual characteristics are of interest.

Although widely used metrics and datasets were employed in the design and implementation of the algorithms in this work, partially in order to provide comparable results with existing systems time limitations allowed for implementation and comparison with two benchmark systems only. Further benchmarks should be implemented in order to provide further comparative results which would validate this work. This is particularly important for the case of the speed studies where research is scarce and the algorithms are sensitive to the test data hence a common point of reference is necessary.

The same applies for the data-sets used throughout this work. Although proof of concept has been validated using these data-sets extensive results with larger and even more diverse content would provide even more concrete performance evaluation for real life applications of the algorithms.

A final study which would further validate the present work would be a formal analysis of the processing cost involved in each algorithm in terms of number of operations. An interesting extension of this study would be to determine the delta between the processing cost of the Direct Feature Extraction technique versus an equivalent implementation of the feature extraction algorithm on JPEG2000.

Further study of the morphological analysis algorithm is required in order to devise a deterministic way for establishing the number of images that can be excluded from the distance calculation process whilst ensuring that the accuracy remains entirely unaffected. A possible approach would be to calculate the distribution of the histogram elements of the entire database in order to establish

the extent of the variation. This however might add considerable processing overhead thereby invalidating the whole point of using the technique which is to speed up the query.

Similarly for the Boundary Conditions technique, the possibility of using an additional low-cost pre-filtering stage in order to determine if it is worth using the boundary conditions or not on a per query basis should be investigated. This could involve a quick check of the first few upper limits with the last few lower limits to determine the difference between them (positive, negative and distance).

The Unified HVD algorithm requires further improvement in order to be directly comparable with the non-rotation invariant algorithms in terms of accuracy. Two directions that could lead to higher accuracy is to incorporate different feature vectors such as the correleation distance, similarly to (Wang, Bovik et al. 2004). Another extension of Unified HVD which has the potential to improve accuracy is to incorporate a multi-resolution descriptor by combining the LWCP descriptor with the Wavelet Isometric Operators paradigm.

Exploration of new fields and applications

All feature extraction algorithms presented in this thesis involve texture. Although texture is proven to have excellent performance especially when only one feature must be used due to processing power limitations, an obvious extension of the present work is to include the remaining low-level feature descriptors in the systems. That is, to fully incorporate both shape and colour. There are potential candidates in wavelet-based image processing research that can serve as ground work for such extensions (Chang and Kuo 1993; Van de Wouwer, Livens *et al.* 1997).

The JPEG2000 compression suite has excellent perceptual preservation characteristics even at very high compression ratios. This is primarily due to the use of wavelet-based codecs. Most of the work presented in this thesis can be easily extended for use directly on JPEG2000 compressed images, while

extensions can be made to fully exploit the internal mechanics of JPEG2000. That would increase the real-life application potential of the presented ideas.

In DWT based codecs, versions of the same image at multiple resolutions are available on demand, originating from the same redundant data source, by decompressing data only up to a certain level. This could be used as a lead for building a retrieval system of variable accuracy. The user could choose between accuracy and speed when performing a query. This way low accuracy but ultra-fast querying could be used in order to eliminate large segments of data and then finer but slower querying could be performed on the remaining data-set. Another factor that could possibly be used to adjust the ratio of accuracy over processing time is the available system resources when results must be acquired within a pre-defined time frame.

Hopefully, the aforementioned future extensions combined with the methodologies and empirical analyses of this work will provide solid ground work for the embedding of the presented technology into commercial retrieval and classification applications.

Future research should focus on improving the accuracy of the content characterization techniques by providing more accurate models of the human visual system. Improvement in accuracy can also be achieved by building tailored application-based systems. However, it is noteworthy that generic fully automated content-based image retrieval is still regarded as a utopia, as it involves numerous and overly complicated parameters for currently available technology (Liu, Zhang et al. 2007). As it is indicated by many (Kherfi, Ziou et al. 2004; Datta, Joshi et al. 2008) a crucial step forward will be for different disciplines such as mathematics, HVS research, database research and application-specific (*e.g.* medical) research to join forces in order to address the yet unresolved issues.

6.8 Final remarks

In modern systems the bottleneck is not always the locally available processing power or the I/O throughput of a system. As it becomes more and more prevalent, network performance plays a decisive role. The reader need not look further than the cases of the explosion in deployments of storage-area-networks and the vast number of digital resources that are interconnected via the internet. Therefore, it becomes evident that performance needs to be taken into consideration when designing systems rather than just relying on Moore's law (Moore 1965) for ever-increasing power, waiting for slow algorithms of today to become applicable tomorrow.

An interesting product announcement which will commoditize large processing power on conventional PCs is the CUDA platform from nVidia (nVidia 2008). The platform provides Application Programming Interfaces for utilizing the very powerful graphics processors for non-graphics rendering related processing when the latter is not required. Given the huge processing power that today's graphics processors convey it will be very interesting to see the first applications of the technology in image processing.

Early works such as (Ahmad, Kiranyaz et al. 2005; Rosenbaum and Schumann 2006; Xing, Hao et al. 2006) indicate that imaging applications for mobile devices is an unchartered research area with vast scope. The bandwidth, processing power, memory restrictions and, perhaps more importantly, the immense fragmentation of software platforms in mobile devices brought the development world and research community back to the early years of desktop computing. A very interesting characteristic of these small devices is the concentration of a plethora of sensory (GPS sensors, Accelerometers, cameras), behavioural (user action monitoring) and semantic (location information) information. This information can be fused to provide semantic and contextual information for image retrieval which is otherwise impossible to obtain in conventional PC environments. Furthermore, despite the leaps in networking and CPU technology, video and image delivery on mobile phones still relies on network side proxy optimization techniques for practical uses (*e.g.* (Bytemobile 2008; Opera 2008)).

Figure 6.1: "My Wife and Mother-in-Law."
An old woman and a young woman appear in this illustration depending on the viewer's point of focus. Anonymous dated German postcard from 1888 depicts the image in its earliest known form. Here excerpted from (IllusionWorks 2008)

The real metric for any automated system claiming intelligence equal to its human counterpart is to pitch one against the other. Although, in many cases evaluation of automated systems can be performed against rigid quantifiable metrics, this is not the case in generic image retrieval. Human visual perception and recognition is a highly complex mechanism (see figure 6.1) that is yet to be fully encoded (assuming that such conjecture is plausible). As CBIR systems are improved and optimized it would be interesting to perform a study on the extent to which the accuracy ratio of automated over user generated test results becomes smaller.

Annex A: Source Code

Implementation Notes

This Annex includes implementations of the algorithms described in this thesis. The implementations have been built on Microsoft Visual Studio 2005 (C++), MATLAB v6.5 (R13b) and Perl v5 on FreeBSD v4.3 although the latter have also been tested on ActivePerl v5 for Windows.

Some of the implementations (Localization Grid, Direct Feature Extraction and Isometry Model) have been constructed around the QccPack image processing library which is publicly available under GPL license at (Fowler 2008). QccPack is a suite of utilities and libraries for compression, coding and quantization of data. QccPack includes an open source implementation of the Set Partitioning in Hierarchical Trees (SPIHT) compression codec which has been used as a building block for the respective implementations for this work. One of the reasons behind the selection of QccPack is the hope that future (more) stable revisions of the algorithms implemented for this work may be submitted for inclusion to the suite.

Localization Grid

```
// ************************************************
// * LOCALIZATION GRID INDEXING RELATED FUNCTIONS *
// ************************************************
// libQccPackSPIHT.c (partial)

static int LGINDEXgetSectorSignificance ( int *subband, int subbandSizeX,
int subbandSizeY, int sector, int grid_size)
     // this function counts the number of significant coefficients
     // of sector (sector) inside subband (*subband)
        // grid_size is the side dimension of the rectangular grid i.e.
grid_size=4 => localization grid is 4x4 etc.
{
  int cx, cy, significance = 0;
  int sectorHsize = subbandSizeX / grid_size;
  int sectorVsize = subbandSizeY / grid_size;
  int offsetV = sector / grid_size;
  int offsetH = sector - (offsetV * grid_size);

  for ( cx = 0; cx < sectorHsize; cx++ )
    for ( cy = 0; cy < sectorVsize; cy++ )
      if ( subband[ ((offsetH * sectorHsize) + cx) + ( subbandSizeX *
((offsetV * sectorVsize) + cy)) ] == 1 )
             significance++;

  return (significance);
}

static int LGINDEXgetGridAverage ( int *subband, int subbandSizeX, int
subbandSizeY, int grid_size)
{
  int c, average, max = 0, min = (subbandSizeX*subbandSizeY);

  for (c = 0; c < (grid_size*grid_size); c++) {// grid_size^2 = total size
of loc grid. multiplication by itself looks bad but compilers tend to
optimize this nicely.
    average = LGINDEXgetSectorSignificance ( subband, subbandSizeX,
subbandSizeY, c, grid_size);
    if (average < min) min = average;
    if (average > max) max = average;
  }
  average = (max + min) / 2;

  return (average);
}

static void LGINDEXgetLocalizationGrid ( int *Lgrid, int *subband[], int
hsize, int vsize, int layers, grid_size)
{
  int sector;
  int grid_element = 0;
  int average;
  int sub, curlayer;
  int vert, horiz;
  vert = vsize / 2;
```

```
  horiz = hsize / 2;

  for (curlayer = 0; curlayer < layers; curlayer++) {
    for (sub = 0; sub < 3; sub++) {
      // now get the average number of sign. coefs per grid-sector inside
the subband
      average = LGINDEXgetGridAverage (subband[(3 * curlayer) + sub], horiz,
vert, grid_size);

      for (sector = 0; sector < 64; sector++) {
       // apparently the order of the subbands is; subband 0 is the HL
subband of the first level
       // of decomposition, and the last subband (3 * layers) is the coarse
LL subband.
       if (LGINDEXgetSectorSignificance (subband[(3 * curlayer) + sub],
horiz, vert, sector, grid_size) >= average)
         Lgrid[grid_element] = 1;
       else
         Lgrid[grid_element] = 0;

       grid_element++;
      } // for (sector)
    }// for (sub)
    vert /= 2;
    horiz /= 2;
  } // for (curlayer)
}

static void LGINDEXgetTextureVector ( int *Lgrid, int *subband[], int hsize,
int vsize, int layers, Real *ext_vector, grid_size)
{
  int sector;
  int grid_element = 0;
  int sub, curlayer;
  int vert, horiz;
  int tmp;
       Real *vectorValue = (Real*) malloc (sizeof(Real) * (layers * 3));

  total = 0; // this is the total of significant coefficients for all
subbands
  vert = vsize / 2;
  horiz = hsize / 2;

  for (curlayer = 0; curlayer < layers; curlayer++) {
    for (sub = 0; sub < 3; sub++) {
    vectorValue[(curlayer * 3) + sub] = 0;
      for (sector = 0; sector < (grid_size*grid_size); sector++) {
       // apparently the order of the subbands is; subband 0 is the HL
subband of the first level
       // of decomposition, and the last subband (3 * layers) is the coarse
LL subband.

             if (Lgrid[grid_element]) {
                   tmp   =   LGINDEXgetSectorSignificance   (subband[(3   *
curlayer) + sub], horiz, vert, sector);
             vectorValue[(curlayer * 3) + sub] += tmp;
                   total += tmp;
```

```
      }
            grid_element++;
      } // for (sector)
    }// for(sub)

    vert /= 2;
    horiz /= 2;

  } // for (curlayer)

  // now divide vector with total number of significant coeffs to obtain
normalised value
  for (tmp = 0; tmp < (3 * layers); tmp++)
    ext_vector[tmp] = vectorValue[tmp]/total;

   free(vectorValue);
}

// ************************************************
// *              END OF LG INDEXING              *
// ************************************************
// ************************************************
// *       LG INDEXING INTERFACE FUNCTION         *
// ************************************************


int INDEXLGFeatureExtraction(QccBitBuffer *buffer,
                             QccIMGImageComponent *image,
                             Real *ext_vector,
                             int *Lgrid,
                             int createLgrid, // TRUE or FALSE
                             int *num_rows,
                             int *num_cols,
                             int num_levels,
                             const QccWAVWavelet *wavelet,
                             const QccWAVPerceptualWeights *perceptual_weights,
                             double image_mean,
                             int max_coefficient_bits,
                             int arithmetic_coded,
                             int *extractionLevel
                                int grid_size)
{
  int return_value;
  QccENTArithmeticModel *model = NULL;
  QccWAVSubbandPyramid subband_pyramid;
  int **sign_array = NULL;
  QccList LSP;
  QccList LIP;
  QccList LIS;
  QccListNode *stop;
  double threshold = 0;//, mythres;
  int row, col;
  int c, d;
  int temphoriz, tempvert;
  int counter;
```

```
  // Create the original significance map
  int  *significanceMap  =  (int  *)  malloc(sizeof(int)  *  ((*num_rows)  *
(*num_cols)));
  int **subband=NULL; //[ num_levels * 3 + 1];
  const                    int                          subbands                    =
QccWAVSubbandPyramidNumLevelsToNumSubbands(num_levels);
  int subbandSize[ subbands];
  temphoriz = (*num_cols) / 2;
  tempvert = (*num_rows) / 2;

  for (c = 0; c < (subbands-1); c += 3) {
    subbandSize[c] = temphoriz * tempvert;
    subbandSize[c+1] = temphoriz * tempvert;
    subbandSize[c+2] = temphoriz * tempvert;
    temphoriz /= 2;
    tempvert /= 2;
    //printf("(feature extraction): during init subbandSize[%d] = %d, [%d] =
%d,   [%d]   =   %d\n",   c,   subbandSize[c],   c+1,   subbandSize[c+1],   c+2,
subbandSize[c+2]);
  }

  subbandSize[subbands-1] = (temphoriz * 2) * (tempvert * 2);

  if ((subband = (int **)malloc(sizeof(int *) * subbands)) == NULL)
    {
      QccErrorAddMessage("(QccSPIHTDecode): Error allocating memory");
      goto QccError;
    }

  for (c = 0; c < subbands; c++)
    subband[c] =(int *) malloc(sizeof(int) * subbandSize[c]);

  for (c = 0; c < subbands; c++)
    for (d = 0; d < subbandSize[c]; d++)
      subband[c][d] = 0;


  vector = (Real *) malloc(sizeof(Real) * subbands);

    if (buffer == NULL)
    return(0);
  if (wavelet == NULL)
    return(0);

  QccWAVSubbandPyramidInitialize(&subband_pyramid);
  QccListInitialize(&LSP);
  QccListInitialize(&LIP);
  QccListInitialize(&LIS);

  subband_pyramid.num_levels = num_levels;
  subband_pyramid.num_rows = (*num_rows);
  subband_pyramid.num_cols = (*num_cols);
  if (QccWAVSubbandPyramidAlloc(&subband_pyramid))
    {
      QccErrorAddMessage("(QccSPIHTDecode):          Error          calling
QccWAVSubbandPyramidAlloc()");
      goto QccError;
```

```
    }

  if ((sign_array = (int **)malloc(sizeof(int *) *(*num_rows))) == NULL)
    {
      QccErrorAddMessage("(QccSPIHTDecode): Error allocating memory");
      goto QccError;
    }
  for (row = 0; row < (*num_rows); row++)
    if ((sign_array[row] =
        (int *)malloc(sizeof(int) * (*num_cols))) == NULL)
      {
       QccErrorAddMessage("(QccSPIHTDecode): Error allocating memory");
       goto QccError;
      }

  for (row = 0; row < (*num_rows); row++)
    for (col = 0; col < (*num_cols); col++)
      {
       subband_pyramid.matrix[row][col] = 0.0;
       sign_array[row][col] = 0;
      }

  if (arithmetic_coded)
    {
      if ((model =
          QccENTArithmeticDecodeStart(buffer,
                                      QccSPIHTArithmeticContexts,
                                      QCCSPIHT_NUM_CONTEXTS,
                                      QccSPIHTArithmeticGetContext,
                                      QCCENT_ANYNUMBITS))
        == NULL)
       {
         QccErrorAddMessage("(QccSPIHTEncode):          Error          calling
QccENTArithmeticDecodeStart()");
         goto QccError;
       }
      QccSPIHTBlockSize = 2;
    }
  else
    QccSPIHTBlockSize = 1;

  if (QccSPIHTAlgorithmInitialize(&subband_pyramid,
                                  &LIP,
                                  &LIS))
    {
      QccErrorAddMessage("(QccSPIHTDecode):          Error          calling
QccSPIHTAlgorithmInitialize()");
      goto QccError;
    }

  threshold = pow((double)2, (double)max_coefficient_bits);

  // ---- MAIN LOOP ----

/*if (extractionLevel > max_coefficient_bits) {
            QccErrorAddMessage("(QccSPIHTFeatureExtraction    ):    Required
threshold must be less than maximum number of bits");
```

```
      goto QccError;
    }*/  //extractionLevel  -  used  for  testing  performance  at  different
extraction levels

//printf("Original  extraction  level  %d,  max  bits  per  coef  %d\n",
extractionLevel, max_coefficient_bits);

  counter = 0; // this keeps the current threshold layer

 for (c = 0; c < (*num_rows) * (*num_cols); c++)
      significanceMap[c] = 0;

  while  (1)//extractionLevel)  //extractionLevel  -  used  for  testing
performance at different extraction levels
    {
      stop = LSP.end;

      // Make sure that the significance map for THIS layer is set to zero;

//     printf("significanceMap is %d big\n",(*num_rows) * (*num_cols));


      return_value =
       QccSPIHTSortingPass(&subband_pyramid,
                           sign_array,
                           buffer,
                           threshold,
                           &LSP,
                           &LIP,
                           &LIS,
                           QCCSPIHT_DECODE,
                           model);
      if (return_value == 1)
       {
         QccErrorAddMessage("(QccSPIHTDecode):          Error          calling
QccSPIHTSortingPass()");
         goto QccError;
       }
      else
       if (return_value == 2){
               break;
         }

      counter++;

      return_value =
       QccSPIHTRefinementPass(&subband_pyramid,
                              buffer,
                              threshold,
                              &LSP,
                              stop,
                              QCCSPIHT_DECODE,
                              model);
      if (return_value == 1)
       {
         QccErrorAddMessage("(QccSPIHTDecode):          Error          calling
QccSPIHTRefinementPass()");
```

```
        goto QccError;
      }
     else
      if (return_value == 2){
        break;
        }

       threshold /= 2.0;
      //printf("Extraction level %d\n", extractionLevel);
   } // while(1)

  return_value = 0;
  goto QccReturn;
 QccError:
  return_value = 1;
 QccReturn:

  // ####################################
  //     FEATURE EXTRACTION PROCEDURE
  // ####################################

  //************************************************************
  INDEXwaveletToSignificanceMap(      &subband_pyramid,      significanceMap,
threshold);
  // Get individual subbands
  INDEXsplitMapToSubbands( significanceMap, subband, *num_cols, *num_rows,
num_levels );
  // if (extractionLevel == counter)
  //   INDEXgetSignificanceMapImage (significanceMap, *num_cols, *num_rows,
image, threshold );

  if (createLgrid)
    LGINDEXgetLocalizationGrid   (Lgrid,   subband,   *num_cols,   *num_rows,
num_levels, grid_size);

  LGINDEXgetTextureVector (Lgrid, subband, *num_cols, *num_rows, num_levels,
ext_vector, grid_size);
  //************************************************************
       *extractionLevel = counter;

  // ---------------------------------
  // Do Feature Extraction housekeeping
  free (significanceMap);

  for (c = 0; c < subbands; c++)
    free (subband[c]);
  free (subband);

  free (vector);

   // -----------------------------
   // Do Qcc Stuff housekeeping

  QccWAVSubbandPyramidFree(&subband_pyramid);
  if (sign_array != NULL)
    {
      for (row = 0; row < (*num_rows); row++)
```

```
        if (sign_array[row] != NULL)
          free(sign_array[row]);
        free(sign_array);
      }
    QccListFree(&LSP);
    QccListFree(&LIP);
    QccListFree(&LIS);
    QccENTArithmeticFreeModel(model);
    return(return_value);
}
```

Pre-filtering type A: Moments & Computational optimization

```
// DistanceCalc.h: interface for the DistanceCalc class.
//
//////////////////////////////////////////////////////////////////////

#if
!defined(AFX_DISTANCECALC_H__4419035F_2338_4E89_B180_1ECF023FE8E3__INCLUDED_)
#define AFX_DISTANCECALC_H__4419035F_2338_4E89_B180_1ECF023FE8E3__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define DISTANCE_MOMENT4 6
#define DISTANCE_MOMENT3 5
#define DISTANCE_MOMENT2 4
#define DISTANCE_EUCLIDIAN2 3
#define DISTANCE_CMOMENT 2
#define DISTANCE_FASTEUCLIDIAN 1
#define DISTANCE_EUCLIDIAN 0

class DistanceCalc
{
public:
	int qind_moment;
	void calcStats();
	void calcMoments(int order);
	void sortMoments(int order);
	void calcAlpha();
	void Euclidian2Distance();
	DistanceCalc ();
	DistanceCalc ( ImageInfoDb *imgdb, ExperimentalSettings *expset,
ImageInfoDb *results, double *dur);
	virtual ~DistanceCalc();

protected:
	int qind;
	Real euclidian1Vector( CSigArray *v1, CSigArray *v2 );
	int numRes;
	void EuclidianDistance();
	void FastEuclidianDistance();
	void momentDistance(int order);
	void momentCombinedDistance();
	void Correlation(); // TO BE IMPLEMENTED LATER USING PEARSON'S FORMULA

	int TYPE;
	ImageInfoDb *imgdb;
	ExperimentalSettings *expset;
	//ExperimentalResults *results; // to be removed
	ImageInfoDb *results;
	double *dur;

	CArray<Real, Real> momentValue;
	CArray<int, int> momentIndex;
};
```

```
#endif //
!defined(AFX_DISTANCECALC_H__4419035F_2338_4E89_B180_1ECF023FE8E3__INCLUDED_)


// DistanceCalc.cpp: implementation of the DistanceCalc class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "Experiment.h"
#include "DistanceCalc.h"
#include "assert.h"

#include "iostream"
using namespace std;
#include "duration.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

DistanceCalc::DistanceCalc()
{
}

DistanceCalc::~DistanceCalc()
{

}

DistanceCalc::DistanceCalc(ImageInfoDb *imgdb, ExperimentalSettings *expset,
ImageInfoDb *results, double *dur) : imgdb (imgdb), expset (expset), results
(results), dur (dur)
{
//     cout << "... Inside DistanceCalc constructor" << endl;

       TYPE = expset->DISTANCE_METHOD;
       numRes = expset->numReturnedResults;
       qind = expset->queryIndex;

//     cout << "... after setting default values" << endl;

       //calcAlpha();
       calcStats();
       calcMoments(2);
       calcMoments(3);
       calcMoments(4);
//     cout << "... after calculating all statistics" << endl;

       //Duration wrapper added at each call
```

```
	CDuration durat;

	durat.Start();
	switch (TYPE) {
	case DISTANCE_EUCLIDIAN:
		cout << endl << "\t\tSlow Euclidian" << endl;
		EuclidianDistance();
		break;
	case DISTANCE_FASTEUCLIDIAN:
		cout << endl << "\t\tFast Distance" << endl;
		FastEuclidianDistance();
		break;
	case DISTANCE_CMOMENT:
		cout << endl << "\t\tCombined Moments" << endl;
		momentCombinedDistance();
		break;
	case DISTANCE_MOMENT2:
		cout << endl << "\t\2nd order Moment" << endl;
		momentDistance(2);
		break;
	case DISTANCE_MOMENT3:
		cout << endl << "\t\3rd order Moment" << endl;
		momentDistance(3);
		break;
	case DISTANCE_MOMENT4:
		cout << endl << "\t\4th order Moment" << endl;
		momentDistance(4);
		break;
	case DISTANCE_EUCLIDIAN2:
		cout << endl << "\t\tEuclidian" << endl;
		Euclidian2Distance();
		break;
	default:
		exit(-1);
	}
	durat.Stop();

	results->expDuration = durat.GetDuration()/1000.0000;
}

void DistanceCalc::EuclidianDistance()
{
	//NOTE RESULTS ARRAY SHOULD BE INITIALIZED BEFOREHAND
	assert(results != NULL);

	CSigArray vecsig = imgdb->getSignature(qind);
	CSigArray othersig;

	// get distance for each picture EXCEPT itself
	for (int c = 0; c < imgdb->getSize(); c++) {
		othersig = imgdb->getSignature(c);
		results->sort(	euclidian1Vector( &othersig, &vecsig),
				c);

	} // for (c)
	results->copySorted(imgdb);
}
```

```
Real DistanceCalc::euclidian1Vector( CSigArray *v1, CSigArray *v2 )
{
	Real distance = 0;

	for (int c = 0; c < v1->GetSize(); c++)
		distance += pow(((v2->GetAt(c)) - (v1->GetAt(c))), 2);

	return distance;
}

void DistanceCalc::FastEuclidianDistance()
{
	CSigArray vecsig = imgdb->getSignature(qind);
	CSigArray othersig;
	Real result = 0;


	for (int c = 0; c < imgdb->getSize(); c++) {
		othersig = imgdb->getSignature(c);
		result = 0;

		for (int i = 0; i < vecsig.GetSize(); i++) {
			result += vecsig.GetAt(i) * othersig.GetAt(i);
		}

		result *= 2;
		results->sort(	(imgdb->getAlpha(c) - result),
					c);
	}
	results->copySorted(imgdb);
}

void DistanceCalc::Euclidian2Distance() // calculates distance using a^2 -
2ab + b^2 instead of (a - b)^2
{
	assert(results != NULL);  //RESULTS ARRAY SHOULD BE INITIALIZED
BEFOREHAND
	CSigArray vecsig = imgdb->getSignature(qind);
	CSigArray othersig;
	Real a, ab, b;
	Real vi, oi;
	for (int c = 0; c < imgdb->getSize(); c++) {   // get distance for
each picture excpt query

		a = ab = b = 0;
		othersig = imgdb->getSignature(c);

		for (int i = 0; i < vecsig.GetSize(); i++) {
			vi = vecsig.GetAt(i);
			oi = othersig.GetAt(i);
			a += oi * oi;
			b += vi * vi;
			ab += oi * vi;
		}
		results->sort(	( a - (2 * ab) + b ),
					c);
	} // for (c)
```

```
	results->copySorted(imgdb);
}

void DistanceCalc::calcAlpha()
{
	Real a;
	CSigArray vecsig;

	for (int c = 0; c < imgdb->getSize(); c++) {
	a = 0;
	vecsig = imgdb->getSignature(c);

		for (int i = 0; i < vecsig.GetSize(); i++)
			a += (vecsig.GetAt(i) * vecsig.GetAt(i));

	imgdb->setAlpha(c, a);

	}// for (c)
}

void DistanceCalc::calcStats()
{
	Real sum, sumSq, mean, var;
	int nData;
	sum = sumSq = 0;
	CSigArray vecsig;

	for (int c = 0; c < imgdb->getSize(); c++) {
		sumSq = 0;
		vecsig = imgdb->getSignature(c);
		nData = int(vecsig.GetSize());

		for (int i = 0; i < nData; i++) {
			sum += vecsig.GetAt(i);
			sumSq += (vecsig.GetAt(i) * vecsig.GetAt(i));
		}// for (i)
		mean = sum / (Real)nData;
		var = sumSq / (Real)nData - square (mean);

		imgdb->setAlpha(c, sumSq); // the "alpha" terminology comes
from fast distance paper
		imgdb->setMean(c, mean);
		imgdb->setVariance(c, var);
	}// for (c)

}

void DistanceCalc::momentCombinedDistance()
{
	Real mmnt_1, mmnt_2, mmnt_3;
	Real moment, cmin, cmax;
	cmin = MaxReal;
	cmax = 0;

	int size, imgdb_size;
	size = imgdb_size = imgdb->getSize();
	size /= 2; // THIS IS WHERE WE DETERMINE HOW BIG IS THE MOMENT SAMPLE
```

```
POOL size /= 2 equals half the database

	// calculate distances for all vectors comprising of 2 3 4th moments
	// sort and do full euclidian
	// for easier implementation - we use the moment=1 to store the
distances

	for (int c = 0; c < imgdb->getSize(); c++) {

		mmnt_1 = pow((imgdb->getMoment(c, 2) - imgdb->getMoment(qind,
2)),2);
		mmnt_2 = pow((imgdb->getMoment(c, 3) - imgdb->getMoment(qind,
3)),2);
		mmnt_3 = pow((imgdb->getMoment(c, 4) - imgdb->getMoment(qind,
4)),2);

		imgdb->setMoment(c,(mmnt_1 + mmnt_2 + mmnt_3),1);
		moment = (mmnt_1 + mmnt_2 + mmnt_3);

		if (moment < cmin) cmin = moment;
		if (moment > cmax) cmax = moment;

	}// for(c)

	// now it can be sorted easily
	sortMoments(1);

	CSigArray vecsig = imgdb->getSignature(qind);
	CSigArray othersig;
	Real result = 0;
	int curr_index;

	// no point to use qind_moment since the query is always first
	for (int c = 0; c < size; c++) {
		curr_index = momentIndex.GetAt(c);
		othersig = imgdb->getSignature(curr_index);
		result = 0;

		for (int i = 0; i < vecsig.GetSize(); i++) {
			result += vecsig.GetAt(i) * othersig.GetAt(i);
		}

		result *= 2;
		results->sort(	(imgdb->getAlpha(curr_index) - result),
					curr_index);

	}
	results->copySorted(imgdb);
}

void DistanceCalc::momentDistance(int order)
{
	int lowerBound, upperBound;
	int size, imgdb_size;
	size = imgdb_size = imgdb->getSize();
	size /= 2; // THIS IS WHERE WE DETERMINE HOW BIG IS THE MOMENT SAMPLE
POOL size /= 2 equals half the database
```

```
	if (qind_moment < (size/2)) { // this makes sure that the window fits
		lowerBound = 0;
		upperBound = size;
	} else if ( qind_moment > (imgdb_size - (size/2)) ) {
		lowerBound = imgdb_size - size;
		upperBound = imgdb_size;
	} else {
		lowerBound = qind_moment - (size/2);
		upperBound = qind_moment + (size/2);
	}

	cout << endl << "\tlowerBnd:" << lowerBound << " upperBnd:" <<
upperBound << " span:" << upperBound - lowerBound << " qind_moment:" <<
qind_moment;

	CSigArray vecsig = imgdb->getSignature(qind);
	CSigArray othersig;
	Real result = 0;
	int curr_index;

	for (int c = lowerBound; c < upperBound; c++) {
		curr_index = momentIndex.GetAt(c);
		othersig = imgdb->getSignature(curr_index);
		result = 0;

		for (int i = 0; i < vecsig.GetSize(); i++) {
			result += vecsig.GetAt(i) * othersig.GetAt(i);
		}

		result *= 2;
		results->sort(	(imgdb->getAlpha(curr_index) - result),
					curr_index);
	}
	results->copySorted(imgdb);
}

void DistanceCalc::calcMoments(int order)
{
	Real moment, total_area;
	CSigArray vecsig;
	Real cmin, cmax;
	cmin = MaxReal;
	cmax = 0;

	for (int c = 0; c < imgdb->getSize(); c++) {
		moment = total_area = 0;
		vecsig = imgdb->getSignature(c);

		for( int i = 0; i < vecsig.GetSize(); i++) {
			moment += pow(float(i),order)*vecsig.GetAt(i);
			total_area += vecsig.GetAt(i);
		}

		moment = moment / total_area;

		if (moment < cmin) cmin = moment;
		if (moment > cmax) cmax = moment;
```

```
            imgdb->setMoment(c, moment, order);
      }// for(c)

      //also need to find what is the index of the query pic wrt to moment
sortlist to establish center
      for (int c = 0; c < imgdb->getSize(); c++)
            if (momentIndex.GetAt(c) == qind) {
                  qind_moment = c;
                  break;
            }
      //cout << "Statistics report: moment min=" << cmin << "  moment max="
<< cmax << endl;
}


void DistanceCalc::sortMoments(int order)
      // so now in order to make the search algorithm fast we create
      // an ordered array of indexes to the pictures wrt the moment
      // note that this sorting only makes sense with single moments since
multi-moment metrics are treated as vectors
{
      Real m_val;
      int size = imgdb->getSize();
      momentValue.SetSize(size+1);
      momentIndex.SetSize(size+1);

      for (int c = 0; c < momentValue.GetSize(); c++)
            momentValue.SetAt(c, MaxReal);

      for (int c = 0; c < imgdb->getSize(); c++) {
            m_val = imgdb->getMoment(c, order);
            for (int i = int(momentValue.GetSize())-2; i > -1; i--) {
                  if (m_val < momentValue.GetAt(i)) {
                        momentValue.SetAt(i+1, momentValue.GetAt(i));
                        momentIndex.SetAt(i+1, momentIndex.GetAt(i));
                        momentValue.SetAt(i, m_val);
                        momentIndex.SetAt(i, c);
                  } else { break; }
                  // if(dist)
            }//for(i)
      }//for(c)

}


// ImageInfoDb.h: interface for the ImageInfoDb class.
//
//////////////////////////////////////////////////////////////////////

#if
!defined(AFX_IMAGEINFODB_H__D81C6040_F586_4332_94CA_B4A64EDB5FDD__INCLUDED_)
#define AFX_IMAGEINFODB_H__D81C6040_F586_4332_94CA_B4A64EDB5FDD__INCLUDED_

#if _MSC_VER > 1000
#pragma once
```

```
#endif // _MSC_VER > 1000

#if !defined(AFX_SIGARRAY_H__E30721AA_F741_41FF_B964_6785974E8F77__INCLUDED_)
#include "SigArray.h"
#endif

class ImageInfoDb
{
public:
	int getIndex( int index );
	void sendPics2Dir(CString dirname);
	ImageInfoDb(CString filename);
	ImageInfoDb (int numResults); // constructor when used for results
	virtual ~ImageInfoDb();
	void dumpContents();
	void sort(Real dist, int index);
	void copySorted(ImageInfoDb *source);

	void setDistance(int index, Real mdistance);
	void setVariance(int index, Real mvariance);
	void setMean(int index, Real mmean);
	void setAlpha(int index, Real malpha);
	void setMoment(int index, Real moment, int order);

	Real getDistance(int index);
	Real getVariance(int index);
	Real getMean(int index);
	Real getAlpha(int index);
	CSigArray getSignature (int index);
	CString getFilename (int index);
	Real getMoment (int index, int order);
	int getSize();

	double expDuration;

protected:
	int parseSignature (CString signature);
	int parseDbFile(CString filename);

	typedef CSigArray signature;
	CArray<Real, Real> alpha;

	CArray<Real, Real> moments[5];
	CArray<Real, Real> mean;
	CArray<Real, Real> variance;

	CArray<signature, signature&> signatures;
	CStringArray filenames;

	CArray<Real, Real> distance; // when used for results
	CArray<int, int> sortIndex;
};

#endif //
!defined(AFX_IMAGEINFODB_H__D81C6040_F586_4332_94CA_B4A64EDB5FDD__INCLUDED_)
```

```
// ImageInfoDb.cpp: implementation of the ImageInfoDb class.
//
//////////////////////////////////////////////////////////////////////
#ifndef __AFXTEMPL_H__
#include <afxtempl.h>
#endif

#include "stdafx.h"
//#include "Fast Distance.h"
#include "iostream"
using namespace std;
#include "assert.h"

//#ifndef __AFXCOLL_H__
//#include <afxcoll.h>
//#endif

#ifndef _GLOBAL_
#include "global.h"
#endif

#include "SigArray.h"
#include "ImageInfoDb.h"

//#include "errordlg.h"
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

ImageInfoDb::ImageInfoDb(CString filename)
{
	parseDbFile(filename);
}

ImageInfoDb::ImageInfoDb(int numResults)
{
	// SET ARRAYS TO INITIAL LENGTHS CAUSE THEY WILL BE USED FOR FUTURE
REFERENCE
	moments[1].SetSize(numResults+1);
	moments[2].SetSize(numResults+1);
	moments[3].SetSize(numResults+1);
	moments[4].SetSize(numResults+1);
	mean.SetSize(numResults+1);
	variance.SetSize(numResults+1);
	signatures.SetSize(numResults+1);
	filenames.SetSize(numResults+1);
	distance.SetSize(numResults+1);
	alpha.SetSize(numResults+1);

	sortIndex.SetSize(numResults+1);
```

```
	for (int c = 0; c < distance.GetSize(); c++)
		distance.SetAt(c, MaxReal);

	expDuration = 0;
}

ImageInfoDb::~ImageInfoDb()
{
}

// User defined methods

int ImageInfoDb::parseDbFile(CString filename)
{
	// ADD CODE TO OPEN FILE AND CONNECT TO CArchive
	CFile f;
	if( !f.Open( (LPCTSTR) filename, CFile::modeRead ) ) {
		#ifdef _DEBUG
			afxDump << "Unable to open file " << (LPCTSTR) filename
<< "\n";
		#endif
		exit( -1 );
	}
	cout << "... file " << (LPCTSTR) filename << " opened succesfully" ;

	CArchive sigArchive( &f, CArchive::load);
	CString currentLine;
	int debugCounter = 0;

	while ( sigArchive.ReadString(currentLine) )
	{
		debugCounter++;

		// NOTE: OVERLOAD SWITCH-CASE?
		if (currentLine == "FILE_NAME") {  //IMPORTANT NOTE: ON NEW
FILE_NAME A NEW ELEMENT IN THE ARRAY IS INITIALIZED
			sigArchive.ReadString(currentLine); // ADD ERROR CONTROL
CODE
			filenames.Add(currentLine);
		} else
		if (currentLine == "TEXT_VECT") {
			sigArchive.ReadString(currentLine);
			parseSignature(currentLine);
			// ADD CODE TO CREATE CArray HERE;
		} else
		if (currentLine == "FAST_COEF") {
			//ignore this entry in this implementation
			sigArchive.ReadString(currentLine);
			alpha.Add(
_ttoi(static_cast<LPCTSTR>(currentLine.Left(currentLine.Find('\t')))) * 10);
// had to divide by 100 to save as text
		} else
		if (currentLine == "MOMENT") {
			sigArchive.ReadString(currentLine);
			moments[2].Add(atof((char *)(LPCTSTR)currentLine));
			moments[3].Add(atof((char *)(LPCTSTR)currentLine));
			moments[4].Add(atof((char *)(LPCTSTR)currentLine));
```

```
		} else
		{
			//ADD ERROR CODE HERE
			goto ERROREXIT;
		} // end of long if/elseif sequence
	} // end of while (sigarchive)
	cout << ", " << debugCounter << " entries read" << endl;
	return 0;
ERROREXIT:
	return -1;
}

int ImageInfoDb::parseSignature(CString signature)
{
	// code to extract one by one the numbers from a space-separated
string (signature)
	CString chunk;
	int index;
	CSigArray sig;

	while (!signature.IsEmpty()) {
		index = signature.Find('\t'); // find the first TAB char
		chunk = signature.Left(index); // copy contents upto this point
		signature.Delete(0, index + 1); // now chop this part plus the
space char
		sig.Add(atof((char *)(LPCTSTR)chunk));
	}

	signatures.Add(sig); // this line adds the whole new signature to the
database entry

	return 0;
}

void ImageInfoDb::dumpContents()
{
	CString outputStr;
	CSigArray sig;
	CString duration;
	duration.Format('%.4f',expDuration);

	cout << endl << "\tDuration of experiment: " << expDuration << "ms" <<
endl;
	cout <<		"\t_____" << endl;

	int counter = 0;
	do {
		cout << "\t" << counter << "\t";

		cout << (LPCTSTR) filenames.GetAt(counter) << "\t";

/*		cout << endl << "CoG:";
		sprintf(buf, "%.3f", centersOfGravity.GetAt(counter));
		cout << buf; */

		cout << distance.GetAt(counter) << endl;
```

```
                counter++;
        } while (counter < getSize());

        // create new error window
}

void ImageInfoDb::sort(Real dist, int index)
{
        // we use the last element as a convenience, to speed up things a bit
        // i.e. we don't have to make a different conditional for the first
step in
        // the sort process.
        // However it is ignored at the end.
        for (int c = getSize() - 1; c > -1; c--) {
                if (dist <= distance.GetAt(c)) {
                        sortIndex.SetAt(c+1, sortIndex.GetAt(c));
                        distance.SetAt(c+1, distance.GetAt(c));

                        sortIndex.SetAt(c, index);
                        distance.SetAt(c, dist);
                }// if (dist)
        }// for(c)
}

void ImageInfoDb::copySorted(ImageInfoDb *sourceDb)
{
        for (int c = 0; c < getSize(); c++) {
                moments[2].SetAt(c, sourceDb->getMoment(sortIndex.GetAt(c),2));
                moments[3].SetAt(c, sourceDb->getMoment(sortIndex.GetAt(c),3));
                moments[4].SetAt(c, sourceDb->getMoment(sortIndex.GetAt(c),4));
                mean.SetAt(c, sourceDb->getMean(sortIndex.GetAt(c)));
                variance.SetAt(c, sourceDb->getVariance(sortIndex.GetAt(c)));
                filenames.SetAt(c, sourceDb->getFilename(sortIndex.GetAt(c)));
                signatures.SetAt(c, sourceDb-
>getSignature(sortIndex.GetAt(c)));
                alpha.SetAt(c, sourceDb->getAlpha(sortIndex.GetAt(c)));
        }// for(c)
}
// Get and Set methods

void ImageInfoDb::setMean(int index, Real mmean)
{
        mean.SetAtGrow(index, mmean);
}

void ImageInfoDb::setVariance(int index, Real mvariance)
{
        variance.SetAtGrow(index, mvariance);
}

void ImageInfoDb::setDistance(int index, Real mdistance)
{
        distance.SetAtGrow(index, mdistance);
}

void ImageInfoDb::setMoment(int index, Real moment, int order)
{
```

```
	moments[order].SetAt(index, moment);
}

void ImageInfoDb::setAlpha(int index, Real malpha)
{
	alpha.SetAt(index, malpha);
}

Real ImageInfoDb::getMean(int index)
{
	if (index < mean.GetSize()) {
		return mean.GetAt(index);
	} else { return -1; }
}

Real ImageInfoDb::getVariance(int index)
{
	if (index < variance.GetSize()) {
		return variance.GetAt(index);
	} else { return -1; }
}


Real ImageInfoDb::getDistance(int index)
{
	if (index < distance.GetSize()) {
		return distance.GetAt(index);
	} else { return -1; }
}

Real ImageInfoDb::getMoment(int index, int order)
{
	if (index < moments[order].GetSize()) {
		return moments[order].GetAt(index);
	} else { return -1; }
}

CString ImageInfoDb::getFilename(int index)
{
	if (index < filenames.GetSize()) {
		return filenames.GetAt(index);
	} else { return CString("-1"); }
}

CSigArray ImageInfoDb::getSignature(int index)
{
	// add some error checking. NOTE: I had problem with returning
something else of CSigArray type
	return signatures.GetAt(index);
}

int ImageInfoDb::getSize()
{
	if (sortIndex.GetSize() > 1) {
		return (int(filenames.GetSize())-1);
	} else {
		return (int(filenames.GetSize()));
```

```
	}
}

Real ImageInfoDb::getAlpha(int index)
{
	if (index < alpha.GetSize()) {
		return alpha.GetAt(index);
	} else { return -1; }
}

int ImageInfoDb::getIndex(int index)
{
	if (index < sortIndex.GetSize()) {
		return sortIndex.GetAt(index);
	} else { return -1; }
}


void ImageInfoDb::sendPics2Dir(CString dirname)
{
	// ++++++++++++++++++++++++++++++
	// CLEAN UP DIR FROM OLD RESULTS
	// ++++++++++++++++++++++++++++++

	CFileFind finder;
	char searchpattern[255];
	BOOL moreFiles;

	// contstruct search pattern XXX\*.pgm
	strcpy_s (searchpattern, 255, (char *)(LPCTSTR)dirname);

	strcat_s (searchpattern, "*.pgm");
	moreFiles  = finder.FindFile(CString(searchpattern));

	while (moreFiles)
	{
		moreFiles = finder.FindNextFile(); //HAS TO GO BEFORE
EVERYTHING CAUSE FIRST RETURN FILENAME IS NULL
		CFile::Remove(finder.GetFilePath());
	}

	// END OF CLEAN UP DIR
	// ++++++++++++++++++++

	CString currFilename;
	CString sourceFilename;
	int temp = 0;
	CFile destFile;
	CFile sourceFile;
	CFileException fileException;
	CFileException fileException2;

	char fser = '1';
	char buffer[16384];
	UINT actual;

	cout << "\t... copying results in " << dirname << " ";
```

```
	for (int c = 0; c < getSize(); c++) {
		sourceFilename = filenames.GetAt(c);

		sourceFilename.Delete(sourceFilename.GetLength()-1, 1);
		// IT SEEMS THAT THERE IS A CHARACTER AT THE END (eol OR tab?)
		// WHICH MESSED UP THE OPEN FUNCTION - VC++ bug?

		if ( !sourceFile.Open( (LPCTSTR)sourceFilename,
CFile::modeRead, &fileException ) )
		{
			cout << "Can't open source file" << endl;
			int err = fileException.m_cause;
		    TRACE( "Can't open file %s, error = %d %s\n",
				filenames.GetAt(c), err,
CFileException::ErrnoToException(err)
				);

		}

		currFilename = dirname;
		currFilename.Format( wchar_t("%spic%3d"),currFilename,c);
		currFilename += ".pgm";

		if ( !destFile.Open( (LPCTSTR) currFilename, CFile::modeCreate
|
	    CFile::modeWrite , &fileException2 ) )
		{
			cout << "Can't open dest file" << endl;
		    TRACE( "Can't open file %s, error = %u\n",
				currFilename, fileException2.m_cause );
		}
		do {
			actual = sourceFile.Read(buffer, 16384);
			destFile.Write(buffer, actual);
		} while (actual == 16384);

		sourceFile.Close();
		destFile.Close();

	} //for(c)
	cout << "done!" << endl;
}
```

Pre-filtering type B: Boundary Conditions / Clustering

```
#######################################################
## sub-routines related to Distance calculation
#######################################################

sub calculateWeights { #custom weights calculation
	foreach $i (0..(($vectorLength-1)/3)-1) {
		$weight[0+($i*3)]=1/(4**($i+1));#note precedence of
multiplication
		$weight[1+($i*3)]=1/(4**($i+1));#uterly unelegant but I think
fast!
		$weight[2+($i*3)]=1/(4**($i+1));
	}
	$weight[$vectorLength-1]=1/4**(($vectorLength-1)/3);
}

sub euclidianQuery {
	my $quimage = $_[0];

	foreach $i (0..@sigref-1) {
		$euclidianScore[$i] = euclidianDistance($quimage, $i);
	}

	@euclidianHash{@filename} = (@euclidianScore);
	@sortedEuclidian = sort by_euclidianDistance keys(%euclidianHash);
	#print out sorted results
	foreach (@sortedEuclidian) {
		print "$euclidianHash{$_} $_\n";
	}
}

sub by_euclidianDistance {
	($euclidianHash{$a} <=> $euclidianHash{$b}) || ($a cmp $b);
}

sub euclidianDistance {
	my $qimage = $_[0];
	my $timage = $_[1];
	my $distance = 0;

	foreach $i (0..$vectorLength-1) {
		##$distance += $weight[$i] * ( abs($sigref[$qimage][$i] -
$sigref[$timage][$i]) ** 2);

		$distance += ( abs($sigref[$qimage][$i] - $sigref[$timage][$i])
** 2);
	}
	$distance = sqrt($distance);

	return $distance;
}

sub fastDistanceMaxQuery {
	my $quimage = $_[0];
```

```
	foreach $i (0..@sigref-1) {
		$fastMaxScore[$i] = $alpha[$i] - $qmax[$quimage] * $beta[$i];
		print "calc. of lower bounds $i : $filename[$i] $alpha[$i] -
$qmax[$quimage] * $beta[$i] = $fastMaxScore[$i] \n";
	}

	@fastMaxHash{@filename} = (@fastMaxScore);
	@sortedFastMaxFNames = sort by_fastMaxDistance keys(%fastMaxHash);
	#print out sorted results
	my $county = 0;
	print "Lower bounds ------ qmax is $qmax[$quimage] ------
Starting...\n";
	foreach (@sortedFastMaxFNames) {
		print "$fastMaxHash{$_} $_\n";
		$sortedFastMaxLBounds[$county++] = $fastMaxHash{$_};
	}
	print "Lower bounds ------------ end.\n";
}

sub by_fastMaxDistance {
	($fastMaxHash{$a} <=> $fastMaxHash{$b}) || ($a cmp $b);
}

sub fastDistanceClusteringQuery {
	my ($quimage, $numHits2Display) = @_;

	#FIRST PASS: first of all we need to obtain the sorted list for Qmin
(lower bound)
	fastDistanceMaxQuery($quimage);

	#the sorted filename list is in @sortedFastMax
	#the distance can be obtained from @fastMaxHash -or- $fastMaxHash{-
filename-}
	# i.e. $fastMaxHash{$sortedFastMax[$c]}

	my $matchesFound = 0;
	##%cluster -- array to hold cluster info i.e. cluster[x] = y, x =
image no, y = lower boundary "cluster".
	my $currentMax = -1;

	$lowerbound = -1;
	#SECOND PASS: for each Qmax (upper bound) find smallest range L <=
distance <= U
	print "Upper bounds ------- qmin is $qmin[$quimage] -------
Starting...\n";
	foreach $i (0..@sigref-1) {
		my $templower;
		$fastMinScore[$i] = $alpha[$i] - $qmin[$quimage] * $beta[$i];
		print "$i : $filename[$i] (upper) $alpha[$i] - $qmin[$quimage]
* $beta[$i] = $fastMinScore[$i] \n";
		$templower = lowerBoundBinarySearch($fastMinScore[$i],
@sortedFastMaxLBounds);
##		print "[debug] set $i, $sortedFastMaxLBounds[$lowerbound] <
$fastMaxScore[$i] < $sortedFastMaxLBounds[$lowerbound+1]\n";
		# if X is on the right of the existing last element and
numHits2Display have not been reached yet, discard
		if ($lowerbound == -1 && $templower > -1) {$lowerbound -
```

```
$templower; next; }

            if ($lowerbound > $templower && $templower > -1) {$lowerbound =
$templower;}

      }
      print "Upper bounds ---------------- end.\n";

      print "Lowerbound found @ $lowerbound\n";

}

sub by_cluster {
      ($cluster{$a} <=> $cluster{$b}) || ($a cmp $b);
}

sub getClusterStats {
      my %hashy = @_;
      my @clusterStats;

      foreach $oh (0..@sigref-1) { $clusterStats[$oh] = 0; }

      while (($key,$value) = each %hashy) {
            $clusterStats[$value]++;
      }

      return @clusterStats;
}

sub getMaxHashKey {
      my %hashy = @_;
      my $maxkey = -1;

      while (($key,$value) = each %hashy) {
            if ($key > $maxkey) {$maxkey  = $key;}
          }

      return $maxkey;
}

sub deleteHashKeyEntries {
      my ($key, %hashy) = @_;
      foreach $key (keys %HASH) {
            delete $HASH{$key};
      }
}

sub lowerBoundBinarySearch {
      my ($lookup, @listy) = @_;

      $size = @listy;
      $ubound = $size;
      $lbound = 0;
      $found = -1;

      if ($lookup > $listy[$size-1]) { return $found; };
```

```
        while ( $ubound >= $lbound ) {
                $median = int( $lbound + (($ubound - $lbound) / 2 ) );
                if ( $lookup == $listy[$median] ) { $found = $median; last; }
                if ( $lookup > $listy[$median] ) { $lbound = $median + 1; next;
};
                if ( $lookup < $listy[$median] ) { $ubound = $median - 1; next;
};
                }

        if ($found == -1) { $found = $lbound - 1 };
        return $found;
}


#######################################################
## sub-routines related to performing the Query
#######################################################

sub performQuery {
        ## get results

        if ($distanceMethod eq 'Euclidian distance') {
                if          (exists          ($sortedEuclidian[0]))          {delete
$sortedEuclidian[0..@sortedEuclidian-1];}
                $t0 = [gettimeofday];
                euclidianQuery($queryImage);
                displayMatches(tv_interval($t0), @sortedEuclidian);
                }
        if ($distanceMethod eq 'Fast distance (clustering)') {
                if          (exists          ($sortedClustering[0]))          {delete
$sortedClustering[0..@sortedClustering-1];}
                $t0 = [gettimeofday];
                fastDistanceClusteringQuery($queryImage);
                displayMatches(tv_interval($t0), @sortedClustering);
                }

        ## display statistics
        ## display results

}
```

Direct Feature Extraction

```
// ************************************************
// * DIRECT FEATURE EXTRACTION INDEXING FUNCTIONS *
// ************************************************
// libQccPackSPIHT.c (partial)

static void INDEXsplitMapToSubbands (int *significanceMap, int *subband[],
int hsize, int vsize, int layers)
{
	int v, h, curlayer;
	int lowvert, lowhoriz, endvert, endhoriz;
	//printf("Update subbands started....\n");
	lowvert = vsize / 2;
	lowhoriz = hsize / 2;
	endvert = vsize;
	endhoriz = hsize;
	//printf("for loops starting...\n");
	for (curlayer = 0; curlayer < layers; curlayer++) {
		for (v = 0; v < (endvert - lowvert); v++) {
			for (h = 0; h < (endhoriz - lowhoriz); h++) {
			//printf("curlayer %d, v %d, h %d, endvert %d, endhoriz
%d, lowvert %d, low horiz %d\n",
			//curlayer, v, h, endvert, endhoriz, lowvert, lowhoriz);

			// the order of the subbands is; subband 0 is the HL
subband of the first level
			// of decomposition, and the last subband (3 * layers)
is the coarse LL subband.
			// HL subband
			subband[3*curlayer][h + (lowhoriz * v)] +=
significanceMap[lowhoriz + h + (lowhoriz * v)];
			// HH subband
			subband[3*curlayer+1][h + (lowhoriz * v)] +=
significanceMap[lowhoriz + h + (lowhoriz * (lowvert + v))];
			// LH subband
			subband[3*curlayer+2][h + (lowhoriz * v)] +=
significanceMap[h + (lowhoriz * (lowvert + v))];
			} // for (h)
		} // for (v)

		endvert = lowvert;
		endhoriz = lowhoriz;
		lowvert /= 2;
		lowhoriz /= 2;
	} // for (curlayer)

	for (v = 0; v < endvert; v++) {
		for (h = 0; h < endhoriz; h++) {
			// LL subband
			subband[3 * layers][h + (endhoriz * v)] +=
significanceMap[h + (endhoriz * v)];
		} // for (h)
	} // for (v)

static int INDEXgetSubbandSignificance (int curr, int *subband[], int
```

```
subbandSize)
{
	int c, result = 0;
	for (c = 0; c < subbandSize; c++)
		result += subband[curr][c];
//	printf("(significant coef): subband [%d] significant %d\n", curr,
result);
	return result;
}

static void INDEXgetTextureVector (int *subband[], int *subbandSize,
				const int subbands, int threshold)
{
	int c;
	total = 0;

	// Get no. of significant coefs for subbands plus the big total
	for (c = 0; c < subbands; c++) {
		subbandSig[c] = 0;
		subbandSig[c] = INDEXgetSubbandSignificance ( c, subband,
subbandSize[c] );
		total += subbandSig[c];
	//	printf("(gettexturevector): subbandSig[%d] = %d\n",c,
subbandSig[c]);
		} // for(c)

	// ... and finally get the normalized vector (and the weights)
	for (c = 0; c < subbands; c++) {
		if ( total > 0 && subbandSig[c] > 0 ) {
			vector[c] = (Real) subbandSig[c] / total;
			}
		else {
			vector[c] = 0;
		}

		if ( subbandSig[c] > total ) {
			printf("error! subbandSig > total", c);
			exit(-1);
			}
	//	printf("(gettexturevector): subbandSize[%d] =
%d",c,subbandSize[c]);
	//	printf("(gettexturevector): vector[%d] = %g, weight[%d] =
%g\n",c, vector[c],c, weight[c]);
		}

}

static void INDEXprintSubbandInfo (QccWAVSubbandPyramid *pyramid, int
*subbandSize, const int subbands, int thres)
{
	int c;

	printf("Subband information : threshold %d , total sign. %d, hsize %d,
vsize %d\n", thres, total, pyramid->num_cols, pyramid->num_rows );
	printf("-------------------------------------------------------------
-------------------\n");
	for (c = 0; c < subbands; c++)
```

```
            printf("  subband  %d:  size  %d,  significant  %d\n",  c,
subbandSize[c], subbandSig[c]);

      printf("\n");
}

static void INDEXwaveletToSignificanceMap(QccWAVSubbandPyramid *pyramid, int
*significanceMap, double threshold)
{
      int row, col;
      double coefficient_magnitude;
      int numofcol, numofrow;
      numofcol = pyramid->num_cols;
      numofrow = pyramid->num_rows;

  for (row = 0; row < numofrow; row++)
    for (col = 0; col < numofcol; col++)
      {
            coefficient_magnitude = fabs(pyramid->matrix[row][col]);

            if ( (numofrow*numofcol) <= col+(row*numofcol) ) {
                  QccErrorAddMessage("(QccSPIHTDecode):  out  of  array
boundaries ");
                  exit(-1);
                  }


            if (coefficient_magnitude > threshold)
              {
                  significanceMap[col + (row * numofcol)] = 1;
              }
            else {
                  significanceMap[col + (row * numofcol)] = 0;
            }
      } // for(col)
}

// ########################################**********************
//           FEATURE EXTRACTION INTERFACE FUNCTIONS
// ########################################**********************
// Interface routine returning one SES texture vector for each threshold
level.
// By exploiting the arithmetic coding of SPIHT feature information is
extrated while reading the compressed bit stream
int INDEXFeatureExtractionSplit(QccBitBuffer *buffer,
                        QccIMGImageComponent *image,
                      Real *ext_vector,
                        int *num_rows,
                        int *num_cols,
                        int num_levels,
                        const QccWAVWavelet *wavelet,
                        const QccWAVPerceptualWeights *perceptual_weights,
                        double image_mean,
                        int max_coefficient_bits,
                        int arithmetic_coded,
                        int *extractionLevel)
{
```

```
  int return_value;
  QccENTArithmeticModel *model = NULL;
  QccWAVSubbandPyramid subband_pyramid;
  int **sign_array = NULL;
  QccList LSP;
  QccList LIP;
  QccList LIS;
  QccListNode *stop;
  double threshold;
  int row, col;
  int c, d;
  int temphoriz, tempvert;
  int counter;

  // stuff to be used for the significant map
       int *significanceMap = (int *) malloc(sizeof(int) * ((*num_rows) *
(*num_cols)));
       int **subband=NULL; //[ num_levels * 3 + 1];
       const                  int                  subbands                  =
QccWAVSubbandPyramidNumLevelsToNumSubbands(num_levels);
       int subbandSize[ subbands];
       temphoriz = (*num_cols) / 2;
       tempvert = (*num_rows) / 2;

       for (c = 0; c < (subbands-1); c += 3) {
              subbandSize[c] = temphoriz * tempvert;
              subbandSize[c+1] = temphoriz * tempvert;
              subbandSize[c+2] = temphoriz * tempvert;
              temphoriz /= 2;
              tempvert /= 2;
              //printf("(feature extraction): during init subbandSize[%d] =
%d, [%d] = %d, [%d] = %d\n", c, subbandSize[c], c+1, subbandSize[c+1], c+2,
subbandSize[c+2]);
              }

       subbandSize[subbands-1] = (temphoriz * 2) * (tempvert * 2);

    if ((subband = (int **)malloc(sizeof(int *) * subbands)) == NULL)
    {
      QccErrorAddMessage("(QccSPIHTDecode): Error allocating memory");
      goto QccError;
    }

       for (c = 0; c < subbands; c++)
              subband[c] =(int *) malloc(sizeof(int) * subbandSize[c]);

       for (c = 0; c < subbands; c++)
              for (d = 0; d < subbandSize[c]; d++)
                     subband[c][d] = 0;


       vector = (Real *) malloc(sizeof(Real) * subbands);
       subbandSig = (int *) malloc(sizeof(int) * subbands);
       //printf("num of levels %d\n", num_levels);
       //printf("num of subbands %d", subbands);

//            printf("%g\n",image_mean);
```

```
  // ---------------------------------------

    if (buffer == NULL)
    return(0);
  if (wavelet == NULL)
    return(0);

  QccWAVSubbandPyramidInitialize(&subband_pyramid);
  QccListInitialize(&LSP);
  QccListInitialize(&LIP);
  QccListInitialize(&LIS);

  subband_pyramid.num_levels = num_levels;
  subband_pyramid.num_rows = (*num_rows);
  subband_pyramid.num_cols = (*num_cols);
  if (QccWAVSubbandPyramidAlloc(&subband_pyramid))
    {
      QccErrorAddMessage("(QccSPIHTDecode):          Error          calling
QccWAVSubbandPyramidAlloc()");
      goto QccError;
    }

  if ((sign_array = (int **)malloc(sizeof(int *) *(*num_rows))) == NULL)
    {
      QccErrorAddMessage("(QccSPIHTDecode): Error allocating memory");
      goto QccError;
    }
  for (row = 0; row < (*num_rows); row++)
    if ((sign_array[row] =
        (int *)malloc(sizeof(int) * (*num_cols))) == NULL)
      {
       QccErrorAddMessage("(QccSPIHTDecode): Error allocating memory");
       goto QccError;
      }

  for (row = 0; row < (*num_rows); row++)
    for (col = 0; col < (*num_cols); col++)
      {
       subband_pyramid.matrix[row][col] = 0.0;
       sign_array[row][col] = 0;
      }

  if (arithmetic_coded)
    {
      if ((model =
          QccENTArithmeticDecodeStart(buffer,
                                      QccSPIHTArithmeticContexts,
                                      QCCSPIHT_NUM_CONTEXTS,
                                      QccSPIHTArithmeticGetContext,
                                      QCCENT_ANYNUMBITS))
        == NULL)
       {
        QccErrorAddMessage("(QccSPIHTEncode):          Error          calling
QccENTArithmeticDecodeStart()");
        goto QccError;
       }
```

```
      QccSPIHTBlockSize = 2;
    }
  else
    QccSPIHTBlockSize = 1;

  if (QccSPIHTAlgorithmInitialize(&subband_pyramid,
                                  &LIP,
                                  &LIS))
    {
      QccErrorAddMessage("(QccSPIHTDecode):          Error          calling
QccSPIHTAlgorithmInitialize()");
      goto QccError;
    }

  threshold = pow((double)2, (double)max_coefficient_bits);

  // ---- MAIN LOOP ----

       counter = 0; // this is the current threshold layer

 for (c = 0; c < (*num_rows) * (*num_cols); c++)
       significanceMap[c] = 0;

  while (1)//extractionLevel)
    {
      stop = LSP.end;

      return_value =       QccSPIHTSortingPass(&subband_pyramid,
                                  sign_array,
                                      buffer,
                                      threshold,
                                      &LSP,
                                      &LIP,
                                      &LIS,
                                      QCCSPIHT_DECODE,
                                      model);
      if (return_value == 1)
              {
                    QccErrorAddMessage("(QccSPIHTDecode):    Error    calling
QccSPIHTSortingPass()");
                    goto QccError;
              }
      else if (return_value == 2){
              INDEXwaveletToSignificanceMap(                  &subband_pyramid,
significanceMap, threshold);
              // Get individual subbands
              INDEXsplitMapToSubbands( significanceMap,  subband,  *num_rows,
*num_cols, num_levels );

              INDEXprintSubbandInfo    (    &subband_pyramid,    subbandSize,
subbands, counter );

              INDEXgetTextureVector(    subband,    subbandSize,    subbands,
counter);
              for (c = 0; c < (subbands-1); c++)
                    ext_vector[c + (counter * (subbands - 1))] = vector[c];
              counter++;
```

```
            break;
        } else if (return_value == 0){
            INDEXwaveletToSignificanceMap(                    &subband_pyramid,
significanceMap, threshold);
            // Get individual subbands
            INDEXsplitMapToSubbands( significanceMap,  subband,  *num_rows,
*num_cols, num_levels );

            INDEXprintSubbandInfo    (    &subband_pyramid,    subbandSize,
subbands, counter );

            INDEXgetTextureVector(    subband,    subbandSize,    subbands,
counter);
            for (c = 0; c < (subbands-1); c++)
                  ext_vector[c + (counter * (subbands - 1))] = vector[c];
            counter++;
      }

    return_value = QccSPIHTRefinementPass(&subband_pyramid,
                           buffer,
                           threshold,
                           &LSP,
                           stop,
                           QCCSPIHT_DECODE,
                           model);

    if (return_value == 1)
       {
         QccErrorAddMessage("(QccSPIHTDecode):          Error          calling
QccSPIHTRefinementPass()");
         goto QccError;
       }
    else if (return_value == 2) {
         break;
       }

    threshold /= 2.0;

    } // while(1)


       *extractionLevel = counter;

  return_value = 0;
  goto QccReturn;
 QccError:
  return_value = 1;
 QccReturn:

       // free feature extraction stuff
       free (significanceMap);

       for (c = 0; c < subbands; c++)
             free (subband[c]);
       free (subband); // SEE BELOW SIGN_ARRAY

       free (vector);
```

```
        free (subbandSig);

        // ------------------------------


  QccWAVSubbandPyramidFree(&subband_pyramid);
  if (sign_array != NULL)
    {
      for (row = 0; row < (*num_rows); row++)
       if (sign_array[row] != NULL)
         free(sign_array[row]);
      free(sign_array);
    }
  QccListFree(&LSP);
  QccListFree(&LIP);
  QccListFree(&LIS);
  QccENTArithmeticFreeModel(model);
  return(return_value);
}
```

Quadtrees

```
static void QTwaveletToSignificanceMap(QccWAVSubbandPyramid *pyramid, int
*significanceMap, double threshold)
{
       int row, col;
       double coefficient_magnitude;
       int numofcol, numofrow;
       numofcol = pyramid->num_cols;
       numofrow = pyramid->num_rows;

  for (row = 0; row < numofrow; row++)
    for (col = 0; col < numofcol; col++)
       {
              coefficient_magnitude = fabs(pyramid->matrix[row][col]);

              if ( (numofrow*numofcol) <= col+(row*numofcol) ) {
                    QccErrorAddMessage("(QccSPIHTDecode): out of array
boundaries ");
                    exit(-1);
                    }


              if (coefficient_magnitude > threshold)
                {
                     significanceMap[col + (row * numofcol)] = 1;
                }
              else {
                     significanceMap[col + (row * numofcol)] = 0;
              }
       } // for(col)
}

int QTFeatureExtraction(QccBitBuffer *buffer,
                        QccIMGImageComponent *image,
                        Real *ext_vector,
                        int *Lgrid,
                        int createLgrid, // TRUE or FALSE
                        int *num_rows,
                        int *num_cols,
                        int num_levels,
                        const QccWAVWavelet *wavelet,
                        const QccWAVPerceptualWeights *perceptual_weights,
                        double image_mean,
                        int max_coefficient_bits,
                        int arithmetic_coded,
                        int *extractionLevel,
                           QTQuadTree *qtvector)
{
  int return_value;
  QccENTArithmeticModel *model = NULL;
  QccWAVSubbandPyramid subband_pyramid;
  int **sign_array = NULL;
  QccList LSP;
  QccList LIP;
  QccList LIS;
```

```
  QccListNode *stop;
  double threshold = 0;//, mythres;
  int row, col;
  int c, d;
  int temphoriz, tempvert;
  int counter;

  // Create the original significance map
  int *significanceMap = (int *) malloc(sizeof(int) * ((*num_rows) *
(*num_cols)));
  int **subband=NULL; //[ num_levels * 3 + 1];
  const int subbands =
QccWAVSubbandPyramidNumLevelsToNumSubbands(num_levels);
  int subbandSize[ subbands];
  temphoriz = (*num_cols) / 2;
  tempvert = (*num_rows) / 2;

  for (c = 0; c < (subbands-1); c += 3) {
    subbandSize[c] = temphoriz * tempvert;
    subbandSize[c+1] = temphoriz * tempvert;
    subbandSize[c+2] = temphoriz * tempvert;
    temphoriz /= 2;
    tempvert /= 2;
  }

  subbandSize[subbands-1] = (temphoriz * 2) * (tempvert * 2);

  if ((subband = (int **)malloc(sizeof(int *) * subbands)) == NULL)
    {
      QccErrorAddMessage("(QccSPIHTDecode): Error allocating memory");
      goto QccError;
    }

  for (c = 0; c < subbands; c++)
    subband[c] =(int *) malloc(sizeof(int) * subbandSize[c]);

  for (c = 0; c < subbands; c++)
    for (d = 0; d < subbandSize[c]; d++)
      subband[c][d] = 0;


  vector = (Real *) malloc(sizeof(Real) * subbands);

    if (buffer == NULL)
    return(0);
  if (wavelet == NULL)
    return(0);

  QccWAVSubbandPyramidInitialize(&subband_pyramid);
  QccListInitialize(&LSP);
  QccListInitialize(&LIP);
  QccListInitialize(&LIS);

  subband_pyramid.num_levels = num_levels;
  subband_pyramid.num_rows = (*num_rows);
  subband_pyramid.num_cols = (*num_cols);
  if (QccWAVSubbandPyramidAlloc(&subband_pyramid))
```

```
    {
      QccErrorAddMessage("(QccSPIHTDecode): Error calling
QccWAVSubbandPyramidAlloc()");
      goto QccError;
    }

  if ((sign_array = (int **)malloc(sizeof(int *) *(*num_rows))) == NULL)
    {
      QccErrorAddMessage("(QccSPIHTDecode): Error allocating memory");
      goto QccError;
    }
  for (row = 0; row < (*num_rows); row++)
    if ((sign_array[row] =
        (int *)malloc(sizeof(int) * (*num_cols))) == NULL)
      {
      QccErrorAddMessage("(QccSPIHTDecode): Error allocating memory");
      goto QccError;
      }

  for (row = 0; row < (*num_rows); row++)
    for (col = 0; col < (*num_cols); col++)
      {
      subband_pyramid.matrix[row][col] = 0.0;
      sign_array[row][col] = 0;
      }

  if (arithmetic_coded)
    {
      if ((model =
          QccENTArithmeticDecodeStart(buffer,
                                      QccSPIHTArithmeticContexts,
                                      QCCSPIHT_NUM_CONTEXTS,
                                      QccSPIHTArithmeticGetContext,
                                      QCCENT_ANYNUMBITS))
        == NULL)
       {
        QccErrorAddMessage("(QccSPIHTEncode): Error calling
QccENTArithmeticDecodeStart()");
        goto QccError;
       }
      QccSPIHTBlockSize = 2;
    }
  else
    QccSPIHTBlockSize = 1;

  if (QccSPIHTAlgorithmInitialize(&subband_pyramid,
                                  &LIP,
                                  &LIS))

    {
      QccErrorAddMessage("(QccSPIHTDecode): Error calling
QccSPIHTAlgorithmInitialize()");
      goto QccError;
    }

  threshold = pow((double)2, (double)max_coefficient_bits);

  // ---- MAIN LOOP ----
```

```
  counter = 0; // this keeps the current threshold layer during recursion

 for (c = 0; c < (*num_rows) * (*num_cols); c++)
      significanceMap[c] = 0;

  while (1)//extractionLevel) //extractionLevel - used only for testing
performance at different extraction levels
    {
      stop = LSP.end;

      return_value =
       QccSPIHTSortingPass(&subband_pyramid,
                           sign_array,
                           buffer,
                           threshold,
                           &LSP,
                           &LIP,
                           &LIS,
                           QCCSPIHT_DECODE,
                           model);
      if (return_value == 1)
       {
         QccErrorAddMessage("(QccSPIHTDecode): Error calling
QccSPIHTSortingPass()");
         goto QccError;
       }
      else
       if (return_value == 2){
               break;
         }

      counter++;

      return_value =
       QccSPIHTRefinementPass(&subband_pyramid,
                              buffer,
                              threshold,
                              &LSP,
                              stop,
                              QCCSPIHT_DECODE,
                              model);
      if (return_value == 1)
       {
         QccErrorAddMessage("(QccSPIHTDecode): Error calling
QccSPIHTRefinementPass()");
         goto QccError;
       }
      else
       if (return_value == 2){
         break;
         }

        threshold /= 2.0;
    } // while(1)

  return_value = 0;
```

```
  goto QccReturn;
 QccError:
  return_value = 1;
 QccReturn:

  //      FEATURE EXTRACTION PROCEDURE

  //*************************************************************
  QTwaveletToSignificanceMap( &subband_pyramid, significanceMap, threshold);
  // Get individual subbands
  QTsplitMapToSubbands( significanceMap, subband, *num_cols, *num_rows,
num_levels );

  // Get significant coefficients at LL subband for seeds
       int lowvert, lowhoriz, endvert, endhoriz;
       int qtcount=0;

       lowvert = vsize / pow(2, layers);
       lowhoriz = hsize / pow(2, layers);
       endvert = vsize / pow(2, layers-1);
       endhoriz = hsize / pow(2, layers-1);

              for (int v = lowvert; v < endvert; v++) {
                     for (int h = lowhoriz; h < endhoriz; h++) {

                     if (significanceMap3 * num_levels][h + (endhoriz * v)])
// if significant coefficient found
                            {
                                   qtcount++;
                                   qtvector[qtcount].LLx = h;
                                   qtvector[qtcount].LLy = v;
                                   qtvector[qtcount].decompLayers =
num_levels; // essentialy this parameter exists only for testing

                                   QTgetSingleQuadtree
(qtvector[qtcount].qtree, significanceMap, *num_cols, *num_rows, num_levels)
                            }
                     } // for h
              } // for v

}

  //*************************************************************

  // ---------------------------------
  // Do Feature Extraction housekeeping
  free (significanceMap);

  for (c = 0; c < subbands; c++)
    free (subband[c]);
  free (subband);

  free (vector);

   // ----------------------------
   // Do Qcc Stuff housekeeping
```

```
  QccWAVSubbandPyramidFree(&subband_pyramid);
  if (sign_array != NULL)
    {
      for (row = 0; row < (*num_rows); row++)
       if (sign_array[row] != NULL)
         free(sign_array[row]);
      free(sign_array);
    }
  QccListFree(&LSP);
  QccListFree(&LIP);
  QccListFree(&LIS);
  QccENTArithmeticFreeModel(model);
  return(return_value);
}

static int QTgetDescendantsSig (
                                                int *significanceMap,
                                                int hsize,
                                                int vsize,
                                                int layer,
                                                int direction, //HL:0, HH:1,
LH:2
                                                int ch,
                                                int cv)
{

       int lowvert, lowhoriz;

       lowvert = vsize / pow(2,layer-1); // we are trying to establish
significance at the _next_ layer
       lowhoriz = hsize / pow(2,layer-1);

              // 4 coefficients per direction
              if (
                     (
                     significanceMap[3*curlayer-3+direction][(ch*2) +
(lowhoriz * (cv*2))]||
                     significanceMap[3*curlayer-3+direction][(ch*2)-1 +
(lowhoriz * ((cv*2)-1))]||
                     significanceMap[3*curlayer-3+direction][(ch*2) +
(lowhoriz * (cv*2))]||
                     significanceMap[3*curlayer-3+direction][(ch*2)-1 +
(lowhoriz * ((cv*2)-1))]
                     )
              return 1;
              else
                     return 0;

}

static void QTgetSingleQuadtree (QTQuadtree *sQTree, int *significanceMap,
int hsize, int vsize, int init_layer)
{
       // sQTree->qtree[direction, node] direction = 0:HL 1:HH 2:LH -- node =
pow(4,layer)+
       int init_layer = sQTree->decompLayers;
       int lowvert, lowhoriz, endvert, endhoriz;
```

```
        int cx, cy, ex, ey;
        lowvert = vsize / pow(2, init_layer);
        lowhoriz = hsize / pow(2,init_layer);
        endvert = vsize / pow(2, init_layer-1);
        endhoriz = hsize / pow(2, init_layer-1);
        cx = (sQTree->LLx);
        cy = (sQTree->LLy);
        ex = 1;
        ey = 1;

        for (curlayer = (layers-1); curlayer > -1; curlayer--)
        {
                for (v = cy-ey; v < cy; v++) {
                        for (h = cx-ex; h < cx; h++) {
                        // the order of the subbands is; subband 0 is the HL
subband of the first level
                        // of decomposition, and the last subband (3 * layers)
is the coarse LL subband.
                        // HL subband:0
                        sQTree->qtree[0,(3*curlayer)*(h+(lowhoriz*v))] =
QTgetDescendantsSig (significanceMap, hsize, vsize, currlayer, 0, h, v);
                        // HH subband:1
                        sQTree->qtree[1,(3*curlayer)*(h+(lowhoriz*v))] =
QTgetDescendantsSig (significanceMap, hsize, vsize, currlayer, 1, h, v);
                        // LH subband:2
                        sQTree->qtree[2,(3*curlayer)*(h+(lowhoriz*v))] =
QTgetDescendantsSig (significanceMap, hsize, vsize, currlayer, 2, h, v);

                        } // for h
                } // for v

                lowvert = endvert;
                lowhoriz = endhoriz;
                endvert *= 2;
                endhoriz *= 2;
                ex =* 2;
                ey =* 2;
                cx =* 2;
                cy =* 2;
        } // for curlayer
}
```

Wavelet Isometry Model

```
static void WIMgetWaveletIsometryModel (QccWAVSubbandPyramid *pyramid, int
*subband[], UnifiedHVD *uHVDvector[], int hsize, int vsize, int angle]
{
	int v, h, curlayer;
	int lowvert, lowhoriz, endvert, endhoriz;
	int HL, LH, HH;
	double s_mean, u_min, u_max;

	numofcol = pyramid->num_cols;
	numofrow = pyramid->num_rows;
	int *dwt_tree_serial [numofcol*numofrow];

	// first serialize wavelet data and discard sign for ease of
manipulation
	for (int row = 0; row < numofrow; row++)
		for (int col = 0; col < numofcol; col++)
		{
			dwt_tree_serial[col + (row * numofcol)] = fabs(pyramid-
>matrix[row][col]);

			if ( (numofrow*numofcol) <= col+(row*numofcol) ) {
			QccErrorAddMessage("(QccSPIHTDecode): out of array
boundaries ");
			exit(-1);
			}
		} // for(col)

	//printf("WIMgetWaveletIsometryModel Update subbands started....\n");
	lowvert = vsize / 2;
	lowhoriz = hsize / 2;
	endvert = vsize;
	endhoriz = hsize;
	umin = MAXREAL;
	umax = 0;

	double radians = cos(angle * (PI / 180));
	double radians2 = cos(2 * angle * (PI / 180));

	//printf("for loops starting...\n");
	for (curlayer = 0; curlayer < layers; curlayer++) {
		for (v = 0; v < (endvert - lowvert); v++) {
			for (h = 0; h < (endhoriz - lowhoriz); h++) {
				//printf("curlayer %d, v %d, h %d, endvert %d,
endhoriz %d, lowvert %d, low horiz %d\n",
				//curlayer, v, h, endvert, endhoriz, lowvert,
lowhoriz);

				// the order of the subbands is; subband 0 is the HL
subband of the first level
				// of decomposition, and the last subband (3 * layers)
is the coarse LL subband.
				// HL subband
				HL = dwt_tree_serial[lowhoriz + h + (lowhoriz *
v)];
				subband[3*curlayer][h + (lowhoriz * v)] +=
(cos(radians) * HL) + (sin(radians) * LH);
				// HH subband
				HH = dwt_tree_serial[lowhoriz + h + (lowhoriz *
(lowvert + v))];
				subband[3*curlayer+1][h + (lowhoriz * v)] +=
(cos(radians2) * HH) + (sin(radians2) * HL) + (sin(radians2) * LH);
```

```
                        // LH subband
                        LH = dwt_tree_serial[h + (lowhoriz * (lowvert +
v))];
                        subband[3*curlayer+2][h + (lowhoriz * v)] +=
(cos(radians) * LH) + (sin(radians) * HL);

                        s_mean += HL + HH + LH;

                        if (u_min > HL) u_min = HL;
                        if (u_min > LH) u_min = LH;
                        if (u_min > HH) u_min = HH;
                        if (u_max < HL) u_max = HL;
                        if (u_max < LH) u_max = LH;
                        if (u_max < HH) u_max = HH;

                    } // for (h)
                } // for (v)

                endvert = lowvert;
                endhoriz = lowhoriz;
                lowvert /= 2;
                lowhoriz /= 2;
        } // for (curlayer)

        s_mean = s_mean / ((hsize*vsize)-(endvert*endhoriz)); // size
excluding LL subband
        uHVDvector->mean = s_mean;
        uHVDvector->max = u_max;
        uHVDvector->min = u_min;
}

static void WIMgetUnifiedHVDVector (int *subband[],  int hsize, int vsize,
UnifiedHVD *uHVDvector)
{
        int v, h, curlayer;
        int lowvert, lowhoriz, endvert, endhoriz;
        int HL, LH, HH;
        double u_min = uHVDvector->min;
        double u_max = uHVDvector->max;

        lowvert = vsize / 2;
        lowhoriz = hsize / 2;
        endvert = vsize;
        endhoriz = hsize;

        int *uHVDhist = (int*) malloc (sizeof(int) * (255)); // 3 * 85
        for (int c = 0; c < 255; c++) {
                uHVDhist[c] = 0;
        }

        //printf("for loops starting...\n");
        for (curlayer = 0; curlayer < layers; curlayer++) {
                for (v = 0; v < (endvert - lowvert); v++) {
                        for (h = 0; h < (endhoriz - lowhoriz); h++) {
                                //printf("curlayer %d, v %d, h %d, endvert %d,
endhoriz %d, lowvert %d, low horiz %d\n",
                                //curlayer, v, h, endvert, endhoriz, lowvert,
lowhoriz);

                        // the order of the subbands is; subband 0 is the HL
subband of the first level
                        // of decomposition, and the last subband (3 * layers)
is the coarse LL subband.
                                // HL subband
                                HL = subband[3*curlayer][h + (lowhoriz * v)]
                                HL = (84)*round(((HL-u_min)/(u_max-u_min)))+1; //
```

```
scale 1-85
                            uHVDhist[HL] += 1;

                            // LH subband
                            LH = subband[3*curlayer+2][h + (lowhoriz * v)]
                            LH = (84)*round(((LH-u_min)/(u_max-u_min)))+86; //
scale 86-170
                            uHVDhist[LH] += 1;
                            // HH subband
                            HH = subband[3*curlayer+1][h + (lowhoriz * v)]
                            HH = (84)*round(((HH-u_min)/(u_max-u_min)))+171;
// scale 171-255
                            uHVDhist[HH] += 1;
                        } // for (h)
                } // for (v)
                endvert = lowvert;
                endhoriz = lowhoriz;
                lowvert /= 2;
                lowhoriz /= 2;

                uHVDvector->uHVDhist[curlayer] = uHVDhist;

                for (int c = 0; c < 255; c++) {
                        uHVDhist[c] = 0;
                }

        } // for (curlayer)

        free (uHVDhist);

}
```

LWCP

```
function out = lwcp_directional(X, orient)
% filename: lwcp_directional.m
% lwcp_directional(X, orient)
%       Creates a binary matrix of subband X using only two bits
%       for characterization one on either side.
% [orient]  defines orientation 1 for horizontal, 2 for vertical,
%           3 for diagonal (\), 4 for diagonal (/)

D = size(X);
sizex = D(2);
sizey = D(1);

Xi = zeros(sizey+2,sizex+2);
Xi(2:sizey+1,2:sizex+1) = X;

%   1   2  (3)
%   4       5
%  (6)  7   8

switch orient
    case 1,% Horiz 4 & 5
        Xi4 = zeros(sizey+2,sizex+2);
        Xi5 = zeros(sizey+2,sizex+2);
        Xi4(2:sizey+1,1:sizex) = X;
        Xi5(1:sizey,1:sizex) = X;
        Xi = (Xi4>=Xi)+2*(Xi5>=Xi);
    case 2,% Vert  2 & 7
        Xi2 = zeros(sizey+2,sizex+2);
        Xi7 = zeros(sizey+2,sizex+2);
        Xi2(3:sizey+2,2:sizex+1) = X;
        Xi7(1:sizey,3:sizex+2) = X;
        Xi = (Xi2>=Xi)+2*(Xi7>=Xi);
    case 3,% Diag  1 & 8
        Xi1 = zeros(sizey+2,sizex+2);
        Xi8 = zeros(sizey+2,sizex+2);
        Xi1(3:sizey+2,3:sizex+2) = X;
        Xi8(2:sizey+1,3:sizex+2) = X;
        Xi = (Xi1>=Xi)+2*(Xi8>=Xi);
    case 4,% Diag  3 & 6
        Xi3 = zeros(sizey+2,sizex+2);
        Xi6 = zeros(sizey+2,sizex+2);
        Xi3(3:sizey+2,1:sizex) = X;
        Xi6(1:sizey,2:sizex+1) = X;
        Xi = (Xi3>=Xi)+2*(Xi6>=Xi);
    otherwise,
        Xi = -1;
end


out=Xi(3:sizey,3:sizex);


% ================================================
% ================================================
```

```
function Sig = processSingleLum(filename)
% filename: processSingleLum.m

filename = filename(1:length(filename)-4);
load([filename,'DWT']);

SigH = [];
SigV = [];
SigD = [];
SigD2 = [];

% the subbands are stored in matrix Y{x}{l}
% where x=1 (LL) 2(LH) 3(HL) 4(HH)
% l= level of decomposition

for i=1:size(Y{2},2)
    lwcpdH = lwcp_directional(Y{2}{i},1);
    lwcpdV = lwcp_directional(Y{3}{i},2);
    lwcpdD = lwcp_directional(Y{4}{i},3);
    lwcpdD2 = lwcp_directional(Y{4}{i},4);

    SigH_t = hist(lwcpdH(:),4);
    SigV_t = hist(lwcpdV(:),4);
    SigD_t = hist(lwcpdD(:),4);
    SigD2_t = hist(lwcpdD2(:),4);

    SigH = [SigH SigH_t];
    SigV = [SigV SigV_t];
    SigD = [SigD SigD_t];
    SigD2 = [SigD2 SigD2_t];
end

Sig = [SigH SigV SigD SigD2];

% Quantized Histogram
% [64 steps]
% ====================
min_Sig = min(Sig);
max_Sig = max(Sig);

Sig = (64-1) * (Sig-min_Sig) / (max_Sig-min_Sig) + 1; %rescale 1-64

tmp = linspace(1,64,64);
[n, Sig] = histc(Sig,tmp); %done quantization


% Histogram Binary Map
% ====================
% thres = threshold
%
% Sig = Sig > thres;_
%


% ================================================
% ================================================
% filename: conv2DWTSubbands(filename, level, filter)
```

```
function conv2DWTSubbands(filename, level, filter)

if (dwtmode('status','nodisp') ~= 'sym')
    dwtmode('sym'); %symmetrical padding
end

%load RGB picture
myRGB = imread (filename);
savefile = filename(1:(length(filename)-4));

savefile = [savefile,'DWT.mat'];

%get YCbCr channels and then separate Luminance Y

if (strcmpi(subString(filename,-3,3),'ras'))
    myYCbCr = myRGB;
    channels = 1;
else
    myYCbCr = rgb2ycbcr(myRGB);
    channels = 3;
end

for i=1:channels

    colourChannel = myYCbCr(:,:,i);

    %WAVELET TRANSFORM

    %convert plane to wavelet toolbox compatible type
    n = 255; % Number of shades in new indexed image
    Ygray = wcodemat(double(colourChannel), n);

    [C,S] = wavedec2(double(colourChannel), level, filter);

    [A] = appcoef2(C,S,'db1',level);
    for dclevel=1:level
        [H{dclevel},V{dclevel},D{dclevel}] = detcoef2('a',C,S,dclevel);
    end

    channelCoefficients{i} = {A H V D};
end

if (channels == 1)
    Y = channelCoefficients(L1id);
    save(savefile,'Y');
else
    Y = channelCoefficients(L1id);
    Cb = channelCoefficients{2};
    Cr = channelCoefficients{3};
    save(savefile,'Y','Cb','Cr');
end
```